An ACT-R Modeling Framework for Interleaving Templates of Human Behavior

Michael Matessa (mmatessa@arc.nasa.gov)
NASA Ames Research Center, Mail Stop 262-4
Moffett Field, CA 94035 USA

Abstract

Performance modeling has been made easier by architectures which package psychological theory for reuse at different levels. Both CPM-GOMS, which packages theory at the task level, and ACT-R, which packages theory at the lower level of rules for perceptual-motor interaction, have been shown to be useful. This paper describes ACT-Stitch, a framework for translating CPM-GOMS templates and interleaving theory into ACT-R. The research involved in producing ACT-Stitch will benefit reusable template research by showing how to implement templates and interleaving in a new architecture that processes resource information. ACT-R research will benefit from re-usable productions packaged at a higher task level and from the multi-tasking control structure used that allows ACT-R to interleave productions from different templates. The zero-parameter predictions of ACT-Stitch are empirically validated.

Introduction

Predicting well-practiced human performance in human-computer interaction (HCI) domains by means of computer modeling is a valuable but difficult process. For example, modeling has been used to predict the outcome of a test of new computer workstations, saving a telephone company millions of dollars per year (Gray, John & Atwood, 1993), but much of the modeling was done by hand.

For accurate predictions, a large amount of psychological theory needs to be applied. Several modeling architectures have been developed to make modeling easier by packaging this theory for reuse. CPM-GOMS (John, 1988; 1990) uses templates of behavior to package at a task level (e.g., mouse move-click, typing) predictions of lower-level cognitive, perceptual, and motor resource use. These templates are interleaved to reflect the ability of skilled people to perform parts of one task in parallel with another. For example, an eye-movement study has demonstrated interleaving in a hand-washing task -- while people perform the subtask of first getting their hands wet they interleave a look to the soap dispenser before performing the motor actions in the subtask of soaping their hands (Pelz & Canosa, 2001). The CPM-GOMS theory has been automated (John, et al., 2002) in a computational architecture that schedules blocks of abstract resource use (Freed et al., 2003). ACT-R (Anderson & Lebiere, 1998; Anderson et al., submitted) uses a computational production system architecture for packaging knowledge at the lower level of rules for working with cognitive and perceptual information and motor actions. In contrast with CPM-GOMS, the ACT-R system can interact with an environment to perceive objects and manipulate them. However, ACT-R does not have a built-in theory of multi-tasking which would interleave tasks, although some work has been done in modeling multi-tasking in the ACT-R architecture (Byrne & Anderson, 2001; Lee & Taatgen, 2002; Salvucci, 2002).

This paper presents a new framework, ACT-Stitch, which combines the usefulness of modeling at the task level with the process theory of a lower-level cognitive architecture. It uses a process of macro-compilation similar to that used by Salvucci and Lee (2003) to translate CPM-GOMS templates into ACT-R productions. Their system will be compared to the current system in the discussion section, but one difference is that their system models at the level of KLM-GOMS, which does not interleave cognitive operators (John & Kieras, 1996). The control structure used by ACT-Stitch to achieve the interleaving of cognitive operators from different templates is one of the major contributions of this paper. The research involved in producing ACT-Stitch will benefit reusable template research by showing what aspects of template and interleaving theory are important in a new architecture that processes resource information. ACT-R research will benefit from re-usable productions packaged at a higher task level and from the multi-tasking control structure used that allows ACT-R to interleave productions from different templates.

Templates

Templates are building blocks of human behavior containing a detailed theory of cognitive, perceptual, and motor behaviors. They are beneficial for modelers because they package this theory at the task level and can be reused in different applications (Matessa et al., 2002). Even behavior as simple as a mouse move and click requires coordination of the use of cognitive, perceptual, and motor resources, as Figure 1 shows in PERT chart form with boxes representing resource use and lines indicating dependencies. The template was developed for the simple task of clicking on lit circles by Gray and Boehm-Davis (2000), but has been successfully reused for clicking to operate a simulated automated teller machine (John, et al., 2002).

Templates require a theory of interleaving to reflect the ability of skilled people to perform operations from different tasks in parallel. When CPM-GOMS was first developed, this interleaving was done by hand, with modelers applying their knowledge of the psychology involved. John et al. (2002) codified this knowledge and implemented automated interleaving in a system that scheduled blocks of abstract resource use. Results from this work were used in the construction of ACT-Stitch templates that produce productions which ACT-R can interleave.

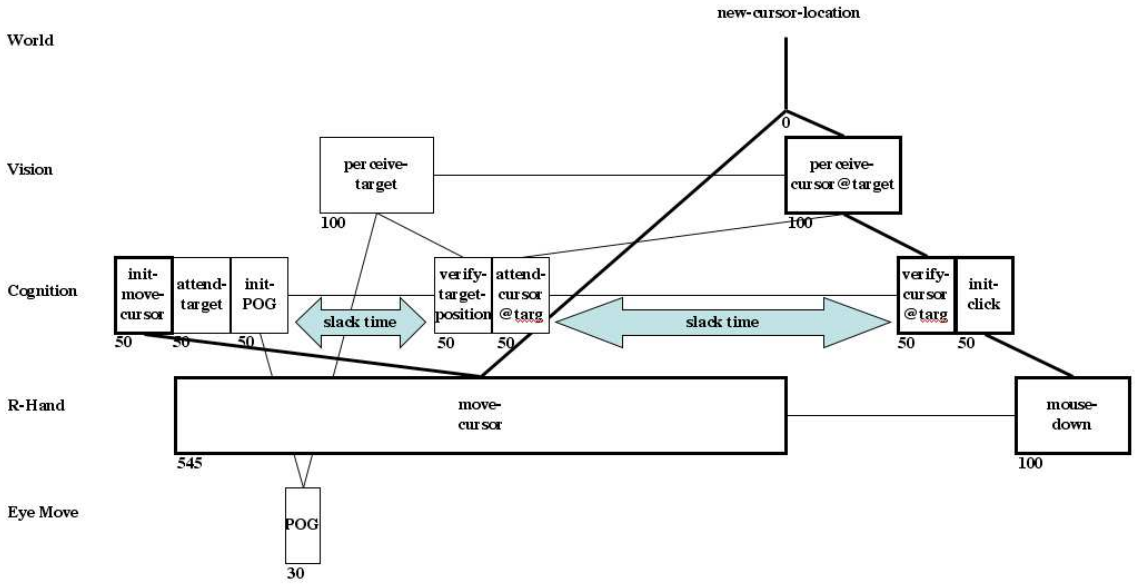

Figure 1: A template of carefully moving the cursor to a target and clicking the mouse (adapted from Gray and Boehm-Davis, 2000).

Macro-Compilation

ACT-Stitch uses a process of macro-compilation to translate CPM-GOMS templates of human behavior into ACT-R productions. More specifically, cognitive operators are translated into productions with ACT-R perceptual-motor commands that represent CPM-GOMS perceptual-motor operators. Productions also contain a control structure that allows ACT-R to implement CPM-GOMS interleaving and have productions from one template execute during the execution of productions from another template. This differs from the ACT-Simple system (Salvucci & Lee, 2003) that compiled a sequence of KLM-GOMS tasks into a series of productions which were controlled by an incrementing state counter.

Macro-compilation should not to be confused with ACT-R production compilation in which two productions are translated into another more efficient production. Salvucci and Lee (2003) argue that macro-compilation facilitates theoretical consistency, inheritance of architectural features, model integration, and model refinement. Theoretical consistency is maintained by having the higher task-level template share a consistent representation with the lower-level ACT-R architecture. The macro-compiled template inherits parameters and limitations that increase psychological plausibility as well as a framework for learning, showing individual differences, and making errors. Model integration is helped by providing a common language where models from different domains can interact.

ACT-Stitch Framework

To understand how ACT-Stitch works, this section will first explain the process of how a modeler uses ACT-Stitch, then describe the ACT-R architecture, then go into more detail about macro-compilation and production execution, and finally give an example of macro-compiled productions.

ACT-Stitch modeling

ACT-Stitch currently has two templates implemented, Slow-Move-Click and Fast-Move-Click, based on templates from Gray and Boehm-Davis (2000). For Gray and Boehm-Davis, Slow-Move-Click represented the selection of a target when there is uncertainty about where the target appears in each trial. Fast-Move-Click represented the selection of a target at a known location. and skipped the verification of the cursor being at the target. These templates were reused by John et al. (2002) in modeling interactions with a simulated automated teller machine. There, Slow-Move-Click represented the selection of difficult targets at far distances, requiring more careful verification of target and cursor location before clicking than the selection of easier targets, which are represented with Fast-Move-Click.

To use ACT-Stitch, the modeler creates two lists, one for target objects and one for a task sequence. The target object list contains target names, positions, and sizes. The task sequence list contains template/target pairs. The system then creates an environment including target objects and macro-compiles templates into productions. The ACT-R system is then run, and information about resource use and

dependencies is automatically stored. This information can be exported to a PERT chart viewing program.

ACT-R

ACT-R (Anderson & Lebiere, 1998; Anderson et al., submitted) is a computational theory of human cognition incorporating both declarative knowledge (e.g., addition facts) and procedural knowledge (e.g., the process of solving a multi-column addition problem) into a production system where procedural rules act on declarative chunks. Chunks are made up of slots containing information, and production rules which match the information in chunk slots are able to execute. The goal chunk represents the current intentions. The ACT-R system includes the capability for modelers to create simulated environments, such as screen interfaces. Production rules have the ability to interact with this environment by perceiving objects and making motor movements through perceptual and motor buffers. With this interaction, ACT-R can make use of Fitts' Law to make predictions of movement time based on distance to target and target size.

ACT-Stitch production creation

CPM-GOMS templates contain predictions of cognitive, perceptual, and motor behavior. When translating a template into ACT-R productions, each cognitive operator in a template corresponds to a production in ACT-R. Cognitive operators and productions are both predicted to take 50 ms to perform by each theory. Both theories predict parallel execution of cognitive, perceptual, and motor processes. In CPM-GOMS, each perceptual and motor operator requires an initiation by a cognitive operator. This corresponds to the ACT-R requirement of productions to initiate vision and motor processes. To move visual attention to a new location and perceive an object, CPM-GOMS predicts that it takes 30 ms to move attention plus some time for perception, while ACT-R predicts that is takes 85 ms to move attention with no additional time for perception. For mouse movement, CPM-GOMS predicts an execution time calculated by Fitts' Law, while ACT-R predicts a 200 ms preparation time plus a time calculated by Fitts' Law plus a 50 ms finish time. For mouse clicks, CPM-GOMS predicts a 100 ms mouse down time plus a 100 ms mouse up time, while ACT-R predicts a 150 ms preparation time plus a 60 ms execution time plus a 90 ms finish time. ACT-R can perform motor preparations in parallel with the motor executions and finishes of previous motor commands, and ACT-Stitch creates productions that take advantage of this capability.

ACT-Stitch creates a set of productions for each template/target pair in the task list, and the productions created from macro-compilation must insure proper sequencing of motor actions, insure the ability to allow the correct productions in future templates to interleave during the execution of productions in the current template, and insure the ability to block the incorrect productions in future templates from interleaving with productions in the current template.

These three requirements are accomplished in productions by using information in the current goal as well as perceptual-motor buffers. Slots in the goal are created for the vision and hand resources for both the intended action and target making use of the resource. This makes four slots in the goal: vision action, vision target, hand action, and hand target. To insure proper sequencing, the action slots in productions of the current template are filled with an intended action appended with the unique number of the current template. Also, the target slots are filled with an intended target. The intended action cannot be used alone since without the template number no sequence information would be stored. The template number cannot be used alone since there may be multiple actions in the same template using the same resource (e.g., mouse move and click). The intended target cannot be used alone since sequence information would be lost if a target appears twice in a sequence (e.g., clicking the same number twice). The intended target cannot be ignored since the same action could be used in a template for two targets (e.g., verify target and verify cursor).

To insure the ability to interleave productions, separate action slots are used for each resource (vision and hand). This allows, for example, a procedure to initiate a vision action from a future template before a procedure initiates a hand action from the current template. To insure the ability to block productions from future templates, the action slots are filled with intended actions appended with the current template number. This prevents, for example, moving to the next target while the hand resource is free between moving to the current target and clicking on the current target. The template number cannot be contained in a separate goal slot because that would not allow productions from the next template to execute before the productions of the current template have finished.

Perceptual-motor buffers are also used in sequencing. Productions that interact with the perceptual-motor buffers check to make sure the buffers are free before using them. Also, the task logic of perception and action makes use of buffers to order productions. For example, the process of verifying a target position before clicking requires filling the visual location buffer with the location of the intended target, then filling the visual object buffer with the object found at that location, and then making a mouse click through the motor buffer.

These goal slots and buffers could be extended to include resources such as a left hand and buffers such as memory retrieval in future template development.

ACT-Stitch production execution

The ACT-R system is initialized with the goal containing the actions and targets of the first template. ACT-R selects productions to execute based on the state of the goal and perceptual-motor buffers. Productions make calls to the perceptual-motor system which has assumptions for how long the resources are used. Slack time corresponds to the time a resource is available during procedure execution. A production that is created from the next template can execute (even if all the productions made from the current

template are not finished executing) when it matches values in the action and target goal slots. Action slots contain intended actions appended with unique template numbers, and target slots contain intended targets. When a resource is no longer needed by a template, a production in the template will fill the action slot with the next intended action appended with the next template number, and the target slot will be filled with the next intended target.

Within-template dependencies are implemented by productions waiting for action and target slots to be filled in the goal and for resources to be available. Template productions are created so that a production will change the contents of action and target slots appropriately. A production (A) from a future template that is waiting for another production (B) in that template to change the contents of action and target slots cannot execute during the execution of productions in the current template until production B is executed.

Relationships across templates are established the same way as within templates, using action and target slots in the goal. Values in these slots allow the blocking of productions that would use resources even if the resource is free.

Example ACT-Stitch productions

To get an idea of what a template looks like after being macro-compiled into ACT-R productions, the following shows pseudo-code for the Fast-Move-Click template. Each instance of a template in the task sequence list would have its own set of productions labeled by the position of the template in the list (x).

```
Tx-Init-Move-Cursor
IF
        right hand action goal is to move in this template
        right hand target goal is this template's object
        motor preparations have completed
THEN
        move cursor
        empty right hand target goal
        set right hand action goal to click in this template

Tx-Attend-Targ
IF
        vision action goal is to attend target in this template
        vision target goal is this template's object
        visual location and object buffers are empty
        vision is available
THEN
        fill visual location buffer with location where
           this template's object should be

Tx-Init-Eye-Move
IF
        vision action goal is to attend target in this template
        vision target goal is this template's object
        visual object buffer is empty
        visual location buffer holds object location
 THEN
        fill visual object buffer with object at location
        empty visual location buffer

Tx-Verify-Targ-Pos
IF
        vision action goal is to attend target in this template
        vision target goal is this template's object
        right hand target goal is empty
        visual object buffer holds object at location y
        location y is the expected location of this template's object
THEN
        empty visual object buffer
        set visual action goal to attend in the next template
        set visual target goal to next template's object
        set right hand target goal to this template's object

Tx-Init-Click
IF
        right hand action goal is to click in this template
        right hand target goal is this template's object
        motor preparations have completed
THEN
        click mouse
        set right hand action goal to move in next template
        set right hand target goal to next template's object
```

Productions that initiate motor movements (Init-Move-Cursor and Init-Click) first check that the motor preparations from previous motor movements have completed Since motor preparations can happen in parallel with motor executions and finishes in ACT-R, this means that preparations can start during previous executions and finishes. Productions could be written to wait for the previous executions and finishes to complete before starting preparations, but they would not be as efficient.

Empirical Validation

ACT-Stitch was applied to the ATM task used by John et al. (2002) to test their automation of CPM-GOMS. The task was to make an $80 withdraw from a checking account on a simulation of an automated teller machine. Users interacted with the ATM by using a mouse to click on simulated keys or slots. The users were instructed to follow the following steps:

Insert card (click on the card slot)
Enter PIN (click on the 4, 9, 0, and 1 keys in turn)
Press OK (click on the OK button)
Select transaction type (click on the withdraw button)
Select account (click on the checking button)
Enter amount (click on the 8 and 0 keys)
Select correct/not correct (click on the correct button)
Take cash (click on the cash slot)
Select another transaction (click on the No button)
Take card (click on the card slot)
Take receipt (click on the cash slot)

This task was repeated 200 times by the users, and results were analyzed using the means of trials 51-100. This level of practice is comparable to that used by both Card, Moran, and Newell (1983) in a text editing task and Baskin and John (1998) in a CAD drawing task when they explored the effects of extensive practice on match to various GOMS models. As in John et al. (2002), Slow-Move-Click templates were used for targets that were difficult to select because of size and distance (e.g. the thin card slot) and Fast-Move-Click templates were used for easier targets (e.g. keypad keys).

Figure 2 compares ACT-Stitch predictions of mouse click times to average subject mouse click times of trials 51-100. The results are highly correlated (r=.96) with a low average absolute difference of 62 ms.

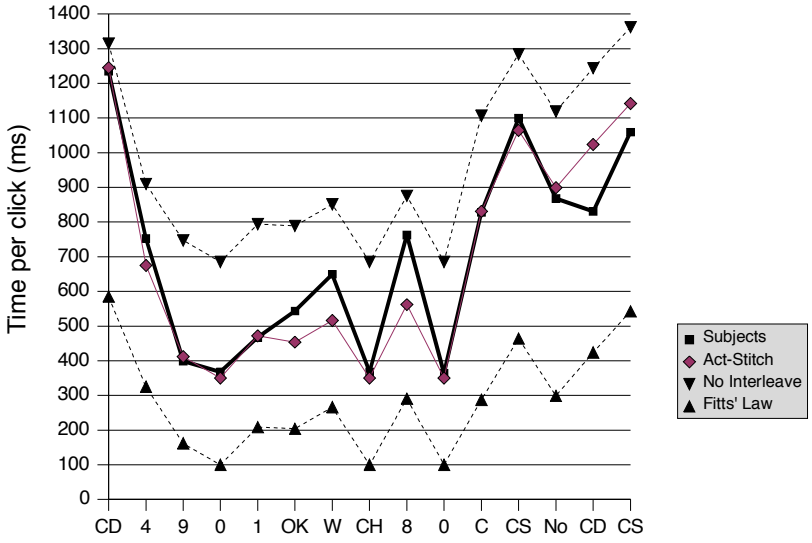

Figure 2: Average subject performance compared to ACT-Stitch predictions, ACT-Stitch predictions with no interleaving, and Fitts' Law predictions.

Figure 2 also shows the value of cognitive modeling over a Fitts' Law only prediction and the value of ACT-Stitch interleaving. A Fitts' Law prediction has a high correlation with subject performance (r=.97) but predicts faster performance, with an average absolute difference of 416 ms. A version of ACT-Stitch was created that did not interleave template productions, and while the correlation with subject performance was still high (r=.95), the predictions are too slow (average absolute distance = 257 ms).

The effect of interleaving on resource use is shown in PERT chart form in Figure 3. This output is from the Sherpa visualization tool developed by John et al. (2002) in their work to automate CPM-GOMS. The top row shows vision resource use, the second shows cognition, the third shows motor preparation, and the bottom shows motor execution and finishing. Resource use is indicated with shaded boxes, and instances of resource use in the same template are shown with the same shade of gray. The figure shows how cognitive, perceptual, and motor resources are interleaved between templates.

General Discussion

ACT-Stitch appears to be a useful framework for modeling the cognitive, perceptual, and motor processes involved in HCI tasks. With a simple description of an environment and task sequence, it is able to produce detailed, zero-parameter predictions that match well to human data.

ACT-Stitch has some similarities and differences with the ACT-Simple framework created by Salvucci and Lee (2003). They both use a process of macro-compilation to translate task-level descriptions of behavior into ACT-R productions, which give a detailed account of the cognitive, perceptual, and motor processes involved in the task. ACT-Stitch adds the ability to easily simulate simple environments, the ability for templates to interleave cognitive operators, and the ability to view resource use of the model with PERT chart tools. With the environment, models can take advantage of Fitts' Law to make detailed predictions of movement times. With a theory of interleaving that is based on fixed resources instead of spontaneous task demands, ACT-R modelers have the ability to start moving away from control theory based on simple chained productions. With PERT chart output, complex interactions of resource use in models can be understood easier.

CPM-GOMS is assumed to model skilled performance, and a CPM-GOMS model translated into ACT-R can be thought of as a state of performance after learning. With the ACT-R compilation process of learning more efficient productions, the whole learning curve from slow reading and remembering instructions to quick interleaving of resources can be studied. There has already been some start on this by Lee and Taatgen (2002), where they describe a model of performance on an air traffic controller task that at first has slow performance to due interpreting instructions, then speeds up due to production compilation creating more efficient productions, and eventually interleaves an optional step to look at wind conditions during multiple keystrokes.

In the ATM task, ACT-Stitch accounts for the data as well as CPM-GOMS automated in another system (see John et al., 2002), but it differs from that system in that it predicts a 200 ms motor preparation that occurs between the movement of attention and motor execution (see Figure 3). ACT-Stitch predicts that during this motor preparation time previous motor operations are taking place. This prediction could be tested with eye-tracking experiments.

This paper offers only a first step of a template and interleaving theory in ACT-R. Many more templates are needed to test the robustness of the representations used for the interleaving theory.

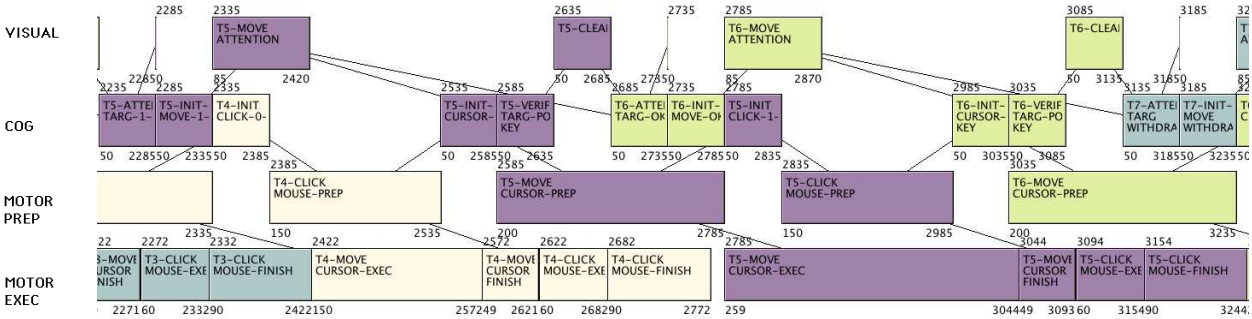

Figure 3: PERT chart of ACT-Stitch interleaving perceptual, cognitive, motor preparation, and motor execution and finishing resources

There are some interleaving abilities that the current framework cannot accomplish, for example, hovering a hand over a key for a key press that occurs in a template that is more than one template away in the future, or blocking an arbitrary combination of resources (such as both hands during typing) from interleaving. But this work is a first step to easier modeling and multi-tasking in ACT-R.

Acknowledgments

This work was supported by Office of Naval Research grant N00014-04-IP-2002 and by funds from the Airspace Operations System project of NASA's Airspace System program.

References

Anderson, J. R., Bothell, D., Byrne M.D. & Lebiere, C. (submitted to Psychological Review). *An Integrated Theory of the Mind*. Available from: http://act-r.psy.cmu.edu/papers/403/IntegratedTheory.pdf

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

Baskin, J. D., and John, B. E. (1998). Comparison of GOMS Analysis Methods. *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems* (Summary) 1998 v.2 p.261-262.

Byrne, M. D., & Anderson, J. R. (2001). Serial modules in parallel: The psychological refractory period and perfect time-sharing. *Psychological Review*, 108, 847-869.

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

Freed, M., Matessa, M., Remington, R. and Vera, A. (2003) How Apex automates CPM-GOMS. *Proceedings of the Fifth International Conference on Cognitive Modeling*, pp. 93-98. Bamberg, Germany:Universitats-Verlag.

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*, 6(4), 322-335.

Gray, W. D., John, B. E. & Atwood, M. E. (1993) Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance. *Human-Computer Interaction*, 8 (3), pp. 237-309.

John, B. E. (1988) *Contributions to Engineering Models of human-computer interaction*. Ph.D. Thesis. Carnegie Mellon University.

John, B. E. (1990) Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. *Proceedings of CHI*, 1990 (Seattle, Washington, April 30-May 4, 1990) ACM, New York, 107-115.

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), pp. 320-351.

John, B. E., Vera, A. H., Matessa, M., Freed, M., and Remington, R. (2002) Automating CPM-GOMS. In *Proceedings of CHI 2002: Conference on Human Factors in Computing Systems* (pp. 147-154). ACM, New York.

Lee, F.J. & Taatgen, N.A. (2002). Multi-tasking as Skill Acquisition. *Proceedings of the twenty-fourth annual conference of the cognitive science society* (pp. 572-577). Mahwah, NJ: Erlbaum.

Matessa, M., Vera, A., John, B., Remington, R., & Freed, M. (2002). Reusable Templates in Human Performance Modeling. *Proceedings of the Twenty-fourth Annual Conference of the Cognitive Science Society* (pp. 649-654). Mahwah, NJ: Erlbaum.

Pelz, J. B. and Canosa, R. (2001). Oculomotor Behavior and Perceptual Strategies in Complex Tasks. *Vision Research*, 41, 3587-3596.

Salvucci, D. D. (2002). Modeling driver distraction from cognitive tasks. *Proceedings of the 24th Annual Conference of the Cognitive Science Society* (pp. 792-797). Mahwah, NJ: Erlbaum.

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