Reusable Templates in Human Performance Modeling

Michael Matessa[1], Alonso Vera[1], Bonnie John[2], Roger Remington[1], and Michael Freed[1]
({rremington, mmatessa, avera, mfreed}@arc.nasa.gov)

[1]Cognition Laboratory, Mailstop 262-4, NASA Ames Research Center
Moffett Field, CA 94035 USA

[2]School of Computer Science, Carnegie Mellon University
Pittsburgh, PA 15213 USA

Abstract

Current computational modeling of human performance can benefit from reusable building blocks of human behavior. Using CPM-GOMS, a cognitively-based task analysis method used in HCI, we have been exploring the concept of reusable templates of common behaviors and their efficacy for generating zero-parameter *a priori* predictions of complex human behavior. This paper details the features we believe are important when moving from hand-crafted models of particular tasks to reusable building blocks of commonly occurring behavior. As this becomes common practice, proportionately more attention can be paid to the task analysis specific to each new domain.

Introduction

To model human behavior successfully one needs to be able to decompose a complex task into a set of primitive operations to which performance parameters may be assigned. These primitives represent the building blocks from which behavior will be constructed such that performance can be predicted for entire task sequences. Thus, the choice of primitives and the method of combining them to construct larger behavior sequences are critical if performance estimates are to be at all accurate. In this paper, we describe a method for combining basic cognitive, perceptual and motor operations into larger behavioral units. These larger units, which will be referred to as *templates*, are applications of psychological theory about the perceptual, motor and cognitive process underlying human performance. Two main points will be argued: 1) task-level reuse is important and 2) behavioral templates are a good way to achieve this reuse. We will present data for a simple HCI task and describe the templates we borrowed (reused) to model it.

Model reuse is a profitable avenue to explore for four reasons. First, consider mousing to a button and clicking on it as an example. People learn and use this skill in most interactions with computers. A model of mousing and clicking on a button should be applicable to many HCI tasks; a new model of mousing and clicking on a button should not be built from scratch for each new task. Second, reuse provides expertise beyond a single researcher. That is, the model for mousing and clicking on a button can be built by researchers with expertise on visual/motor behaviors and the complex model-builder can benefit from that expertise. Third, reuse provides external verification of the component models. If the model for mousing and clicking on a button predicts the behavior well in the context of a complex task, the data provides an independent test of the mechanisms of that model. Finally, reuse provides additional constraint on models of complex tasks. If the mousing and clicking on a button model predicts the behavior well, the HCI modeler should not change the basic mechanisms of the model simply to make it work in a new domain.

Cognitive architectures such as ACT-R (Anderson & Lebiere, 1998) and Soar (Newell, 1990) embody a large set of reusable constraints on behavior prediction. These, however, are mostly at the cognitive level rather than at a task level. We know how to reuse architectures but not content. Modeling performance on human-computer interaction (HCI) tasks is valuable but currently suffers from several problems. Models of task performance need to be handcrafted and typically take a long time to be created. Neither whole models nor their component parts tend to be reused, allowing little transfer of code from one task model to the next. It is difficult to incorporate psychological knowledge into models and cognitive/psychological expertise is needed to do so. Once a model is complete, it may be uninformative with respect to a next attempt at modeling.

Other disciplines that build complex systems, like engineering and computer science, have successfully employed reuse as an approach to tame complexity and this is an approach that cognitive science should explore as well. There have been numerous arguments about the benefits of using a unified cognitive architecture to provide power, structure and constraint (e.g., Anderson, 1983; John & Altmann, 1999; Newell, 1990), but fewer efforts to incorporate previously-built models of generalized capabilities into models of more complex tasks (e.g., Nelson, Lehman & John, 1994).

Cognitive Modeling Methods

Cognitive architectures such as ACT-R and Soar have attempted to solve some of these problems. Underlying mechanisms embody the psychological theories of the architectures, and so the theories do not have to be explicitly coded for each model. Attempts have been made to reuse models at the task level, but in general this is not a widely adopted practice (e.g., John & Lallement, 2000; Nelson, Lehman & John, 1994). As a result, models are mostly handcrafted and take a long time to create.

The field HCI uses cognitive models in several ways. It often is less interested in the process of modeling than in getting results quickly, and has therefore put emphasis on modeling frameworks that are easier to learn and use than the more complex cognitive architectures like Soar and ACT-R (John, 1998). In addition, that field has sought modeling procedures that reuse actual pieces of models as well as the framework. For instance, GOMS models (Card, Moran & Newell, 1983) are constructed by hierarchical goal decomposition with reusable, empirically-determined execution times assigned to particular goals.

The GOMS methodology has proven highly successful in predicting task completion times for skilled users in routine human computer interaction (HCI) tasks (e.g., Gray, John, & Atwood, 1993; John, Vera, & Newell, 1994). GOMS is really a family of analysis techniques in which performance predictions emerge by combining a task decomposition with estimates of completion times for steps in the decomposition. The task decomposition produces a representation of the task as a set of nested goal states that include an initial state and a final goal state. The user is assumed to move from one goal state to another by applying operators that represent actions, such as moving a mouse or reading a word. The set of nested goal states often resembles a hierarchy, but need not form a strict hierarchy. The iterative decomposition into goals and nested subgoals can terminate in leaf nodes (primitives) of any desired granularity, the choice of level of detail dependent on the predictions required. Times are assigned to the operators that transition between goal states, with additional times often assigned to subgoal completion (Kieras, 1994). Since GOMS is meant to model routine behavior, the user is assumed to have methods that apply sequences of operators and subgoals to achieve a goal. Selection rules are applied when there is more than one method to achieve a goal.

The CPM-GOMS extension to GOMS (John, 1990) adds psychological knowledge to the GOMS goal decomposition by expressing common HCI tasks (e.g. reading from a screen, typing) as patterns of Model Human Processor (MHP; Card, Moran & Newell, 1983) operations of its cognitive, perceptual, and motor processors. These patterns, or templates, can help cognitive modeling by grouping psychological knowledge into reusable chunks that can be organized into larger behavioral sequences. Although approaching modeling with reusable templates has been taught for a decade (John & Gray, 1992), it has not become widespread, possibly because CPM-GOMS was not in computational form (it had to be done by hand by drawing PERT charts) until recently (John, Vera, Matessa, Freed & Remington, 2002).

Template Structure

Templates are reusable applications of psychological theory that describe short behavioral sequences. Thirteen templates were offered to CPM-GOMS modelers in tutorials and classes in the early 1990s (John & Gray, 1992) and others have been added since then. One example of a CPM-GOMS template for mouse clicking (created by Gray & Boehm-Davis, 2000) can be seen in Figure 1. The template incorporates a psychological theory of the cognitive, perceptual, and motor components of mouse clicking and dependencies between these components. The template was developed in the context of a simple task of clicking on lit circles, but has successfully been reused in the context of clicking to operate a simulation of an automated teller machine (John, et al., 2002).

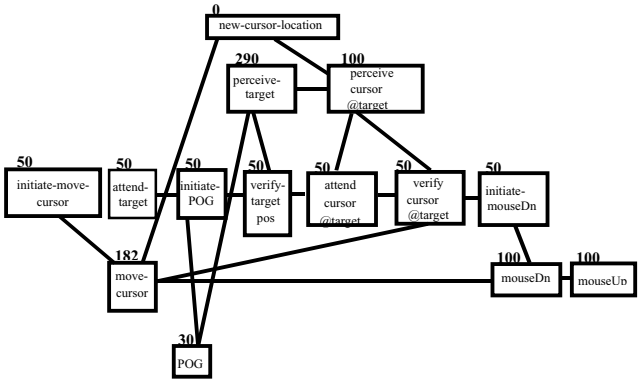

Figure 1: Model of carefully moving the cursor to a target and clicking the mouse button (adapted from Gray & Boehm-Davis, 2000).

The psychology in this template is in the form of the durations, the cognitive bottleneck, the logical dependencies, and the task dependencies. The *durations* for the cognitive, perceptual, and motor components of the template are empirically derived MHP-level estimates of these activities. CPM-GOMS assumes a *cognitive bottleneck,* where cognitive processes cannot execute in parallel. The cognitive resource stream is therefore serially scheduled based on the logical and task dependencies as described below.

According to CPM-GOMS, each motor activity must be preceded by a cognitive activity that initiates it. This is an example of a *logical dependency* — a motor action cannot take place unless a cognitive motor initiation activity has occurred. That you cannot click on a target

until you have moused over to it is an example of a *task dependency*. It is not a CPM-GOMS level logical dependency nor is it a necessary consequence of the cognitive bottleneck; it is true because the task requires it to be so. So, for example, a possible performance error might be to click before the mousing movement to the target is complete. In contrast, executing a motor action without a preceding cognitive initiation is not a possible performance error. Similarly, executing two cognitive activities in parallel is not a possible performance error. This combination of constraints embodies the unique application of psychological theory to the crafting of each template.

Interleaving Templates

Embodying psychological theory in separate templates is only the first step. The challenge is that CPM-GOMS templates need to be interleaved to fully capture the time course of behavior. Simply adding up the performance time predicted by a sequence of templates produces a time that is longer than that of human performance. A key CPM-GOMS assumption is that humans are able to perform certain components of the templates not in strict sequence but interleaved so that some components of a later template can occur in an earlier template.

A concrete example of this interleaving can be seen in a task as simple as hand-washing. While most eye fixations are related to immediate actions such as turning on the faucet, a small number are made to objects relevant only to future actions. Pelz and Canosa (2001), with the aid of an eye tracker, observed that subjects consistently made eye movements to the towel during earlier parts of the hand-washing process.

Part of the psychological theory embedded in templates is the knowledge of what components of one template can occur in parallel with components of another template. This interleaving theory has until recently only been implemented in paper-and-pencil. Details of how interleaving is implemented in our system are presented later. When cognitive components have a dependency on perceptual or motor components which take a relatively long time to complete, slack time may occur when the cognitive resource is not being used. Interleaving involves filling up this slack time with activity from the cognitive components of the next template.

ATM Study & Data

CPM-GOMS has been demonstrated to make accurate zero-parameter *a priori* predictions of skilled HCI behavior. Using templates, we created a CPM-GOMS model of a simple HCI task — withdrawing money from an ATM. We gave two users extensive practice with this task because CPM-GOMS models are expected to predict the performance of highly-skilled users (Baskin & John, 1998).

The ATM Task

The task was to make an $80 withdraw from a checking account on a Visual Basic 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 picture of the card slot)
Enter PIN (click on the 4, 9, 0, and 1 buttons in turn)
Press OK (click on OK button)
Select transaction type (click on withdraw button)
Select account (click on checking button)
Enter amount (click on 8 and 0 buttons)
Select correct/not correct (click on correct button)
Take cash (click on the picture of the cash slot)
Select another transaction (click on No button)
Take card (click on the picture of the card slot)
Take receipt (click on the picture of the cash slot)

This task was repeated 200 times by the users. 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.

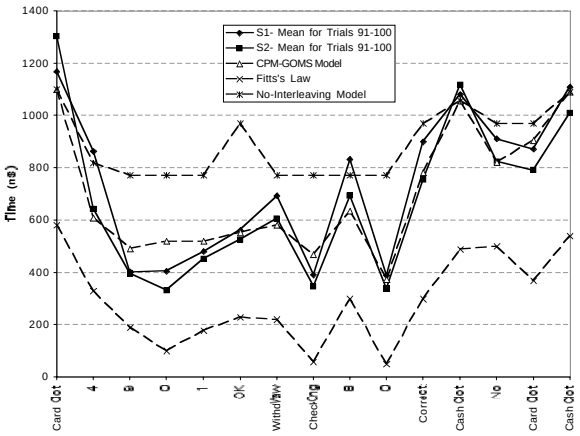

Figure 2: Click times for users and models

The CPM-GOMS Model

The CPM-GOMS model was created by first expressing the hierarchical goal structure of the ATM task. At the bottom of the hierarchy were two CPM-GOMS templates: Slow-Move-Click and Fast-Move-Click. The cognitive, perceptual, and motor components of these templates were taken directly from Gray and Boehm-Davis (2000) where predictions from these templates were compared to data from several variations of a simple target selection task.

The Slow-Move-Click template is shown in PERT chart form in Figure 1. Because there was uncertainty about where a target would appear in each trial, Gray and Boehm-Davis considered Slow-Move-Click to represent a careful selection of a visible target. Fast-

Move-Click represented a more confident selection of a target when the user knew where the target was to appear. In our model, we choose to use Fast-Move-Click for clicking on the ATM buttons because they were a reasonable size and Slow-Move-Click for clicking on the card and cash slots because these slots were thinner and difficult to hit unless the user was careful.

Comparing the Model to Data

Figure 2 shows a comparison of the CPM-GOMS model predictions of mouse click times and mean user click times. Because Baskin and John (1998) have found that CPM-GOMS models predict behavior well at around the 100th trial of a practiced procedure, the means of trials 91-100 for each ATM user are shown. To see the benefit of interleaved templates, predictions from two other models are also shown: a model using only Fitts s Law predictions (a motor time prediction based only on target geometry), and the CPM-GOMS model with sequential templates but no interleaving.

The first thing to notice is the good *a priori* fit of the model to user data. The degree to which interleaving contributes to this fit can be seen by comparing the interleaving and non-interleaving models. The *non-interleaving model* generally predicts a longer time for a mouse click and does not capture the variation of user click times. This variation is better captured by the *Fitts s Law only* model, but the model does not represent the perceptual and cognitive processes incorporated into templates and so predicts faster click times than users produce. These comparisons show that templates contain important predictions of perceptual, cognitive, and motor processes, and that the theory of template interleaving can capture the abilities of users to save time by performing parallel processes.

The important point to emphasize is that the templates used to make these predictions were developed for a very different task and were successfully reused in the present model. Also, note that the tool we have developed to implement CPM-GOMS, Apex (Freed, 1998b, Freed & Remington, 2000), allows us to easily generate alternative versions of the model (e.g., just Fitts s Law, templates with no interleaving). We will expand on the workings of the tool below.

Implementing Templates

How does an organism organize its resource allocation over an extended period of time over an extensive sequence of behaviors? Hierarchical task decomposition is a convenient formalism for an analyst to record and communicate their analysis of a task but it also seems to have some psychological reality in how people organize their environment and cognitive resources to approach a complex task. Many representations of human behavior use hierarchical task decomposition, from Scientific Management at the turn of the century, to HTA (Kirwan and Ainsworth, 1992) to GOMS, Soar and ACT-R. However, in these latter architectures, the hierarchical task decomposition bottoms out at the level of operations of the underlying cognitive, perceptual and motor processors.

It is relatively easy to perform a hierarchical task analysis for each new task domain, and necessary because each domain has its own objects, tasks, knowledge, and procedures. It is much more difficult to describe the cognitive, motor, and perceptual processes underlying every new task. However, many tasks bottom out at the same component behaviors constrained by human abilities. The component behaviors are actions like visual search, mouse movements, and typing. These types of actions are perfect candidates for re-use because they occur repeatedly in many tasks, and must be realized in operations closer to the architecture than is necessary to describe the task domain. Removing the burden of understanding and programming in the underlying cognitive architecture in the form of reusable templates could make cognitive modeling more accessible to a wider range of domain experts.

What does it take to create templates in a cognitive architecture?

- A systematic relationship between the bottom level of the hierarchical task decomposition, the templates, and their realization in the underlying architecture
- A systematic relationship at the boundaries between templates realized in the underlying architecture. This may be more complicated than simple serial execution, as we will describe in the context of CPM-GOMS, the Model Human Processor, and its embodiment in Apex.

Template Grain Size

Generalizeable behavior templates should exist at different grain sizes. From a practical modeling perspective, it is valuable to represent basic HCI behaviors in different size chunks. For example, one branch of a hierarchical goal decomposition may terminate at simply clicking on an item whereas another might terminate at a more complex action such as choosing an item from a pull-down menu. The latter action encompasses the first (twice, in fact), but both are useful as separate templates. Along with generalizability, a valuable constraint on the grain size of templates is the level at which hierarchical goal decomposition would bottom out. For example, a template that is just the reaching component of typing a key (i.e., putting your finger over it without pressing on it) would be very general; it would be Fitts s Law plus the associated cognitive and perceptual components. However, it would not typically be where a hierarchical goal decomposition would bottom out. A hierarchical goal decomposition is more likely to bottom out with activities such as click-on-button or press-key than just

the reaching component of these activities, and templates should reflect this.

Implementing Interleaving

The template-builder gets to deliberately program these dependencies. The relationship between templates in CPM-GOMS involves not sequencing, but interleaving of parallel operators on multiple resources. In highly skilled behavior, people attain a high degree of parallelism in their behavior. To illustrate this point, think about printing a document in the word processor you regularly use. It is not uncommon for people to select the Print command then move the mouse to the place where the OK button will appear well before the dialog box is visible. A hierarchical task analysis would list the steps (1) select Print , (2) wait for dialog box, and (3) click on OK , but the realization in the underlying architecture should allow the mouse movement associated with step (3) to precede the completion of step (2), whereas the clicking associate with step (3) must indeed wait for the completion of step (2) or the task will not be successfully completed. Thus, the relationship between templates is a complex system of interleaving of the architectural-level operations.

Apex allows manipulation of serial and parallel resource in exactly the form required by CPM-GOMS. Apex (Freed, 1998a) has a flexible resource management system that allowed the implementation of the constraints necessary to produce CPM-GOMS template interleaving. A complete description of the implementational rules guiding interleaving in CPM-GOMS as embodied in Apex is available elsewhere (Berkovich & Kwong, 2002; John et al., 2002). In general, we use three facets of the Apex architecture:

1. resource modules (i.e., cognition, hands, point of gaze, visual perception),
2. priorities on the templates, and
3. virtual resources to record a template s intention to use a resource.

Resource modules. To accomplish templates, operators consume resources at the architectural level. The resource modules act serially within themselves, so when they are occupied by an operator, other operators that require that resource must wait until that resource is free again. When more than one operator requires the same resource at one time, there needs to be a way to resolve the conflict and assign that resource to only one operator. Within a template, such conflicts are either avoided because of logical dependencies between the operators (e.g., perceiving a target cannot happen until the eyes have moved to that target) or resolved randomly because the template-builder has determined that it does not matter for the completion of the task whether one goes before the other. When there is contention for resources between operators in different templates, an additional mechanism for conflict resolution is needed.

Priorities. When there is competition for a resource between templates, priorities of the templates is used. Each template is assigned a priority associated with its order in the hierarchical task decomposition. In the printing example above, selecting the Print command (step 1) has the highest priority, waiting for the dialog box (step 2) has the next highest priority, and clicking the OK button (step 3) has the lowest priority. Both steps 1 and 3 will contend for the right hand to move the mouse to their respective targets, and step 1 will win by its priority, ensuring that the print command is selected before the mouse is moved to where the OK button will appear. However, step 2 (waiting) does not require the right hand, so the mouse is free to move to the location of the OK button before the dialog box appears.

Virtual resources. However, resources and priorities are not enough. For instance, the template to select the print command (step 1) requires the eyes to move to and perceive the menu title, and later within the same template to move to and perceive the desired menu item. However, while the hand is completing the move and click to the menu title and the menu is coming up, the eye resource is free. Therefore, step 3, clicking on the OK button wants the eye to move to the location of the OK button, and since the resource is free, there is no contention and priorities do not come into play. Thus, the eye resource could be assigned to the OK button even though the menu item will be appearing in a fraction of a second and the eye will be needed there. Therefore, to reserve the eye for the higher-priority template (in this case, the select-menu-item(Print)), we developed the idea of virtual resources. Virtual resources are analyst-defined entities that act like regular resources. Operators consume these resources and they have to be allocated using priorities. So at the beginning of the select-menu-item (Print) template s first eye movement, it reserves the eye-block virtual resource and only releases that virtual resource after its last visual perception is complete. Since all eye-movements and visual perception require this eye-block virtual resource, this technique blocks any lower-priority template from stealing that resource before a higher-priority template is finished with it.

Discussion

We have a computational system that implements and automatically interleaves reusable behavioral templates. While only two templates have been presented here in detail (Slow-Move-Click and Fast-Move-Click), several others have been implemented in our system for touch-typing and applied to a different computer aided design task, and several more are under development for interacting with the flight maintenance system found on commercial airliners. Previous work has shown that other currently existing templates in the literature are useful for simulations done by hand (Gray, John, &

Atwood, (1993) being a good example). We will work to implement these as well. As more templates are accumulated, issues such as coverage of possible behaviors can be addressed.

We are currently exploring the possibility of generating CPM-GOMS models from ACT-RPM. ACT-RPM imposes its own set of constraints on cognitive, perceptual and motor resources, many of them different from those of CPM-GOMS. This makes the generation of genuine CPM-GOMS behavior from ACT-RPM more challenging but also promises the possibility of extending the predictive scope of CPM-GOMS to a wider range of skill (i.e., from novice to expert) because of ACT-R s new production compilation mechanism (Taatgen & Lee, submitted)

In order for cognitive modeling to come into wider use in the design process, it is necessary to package the abundance of data on human perceptual, cognitive, and motor phenomena into a set of behavioral templates that can be directly incorporated into predictive, computational models. Templates reduce the amount of psychology and modeling methodology required to build models, compile the human performance data into templates, and allow the modeler to focus on task analysis.

Acknowledgments

This research was supported by funds from the NASA Aviation Operations Safety Program and the Intelligent Systems Program.

References

Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. & Lebiere, C. (1998). *The Atomic Components of Thought*. Lawrence Erlbaum Associates.

Baskin, J. D., & 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.

Berkovich, M., J., & Kwong, E. (2002). Apex template manual, working paper.

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

Freed, M. (1998a) Managing multiple tasks in complex, dynamic environments. In Proceedings of 15th National Conference on Artificial Intelligence, (Madison, Wisconsin,) Menlo Park, CA: AAAI Press/ MIT Press. pp. 921-927.

Freed, M. (1998b) *Simulating Human Performance in Complex, Dynamic Environments*. Doctoral Dissertation, Northwestern University.

Freed, Michael and Remington, R. (2000) GOMS, GOMS+ and PDL. *In Working Notes of the AAAI Fall Symposium on Simulating Human Agents*. Falmouth, Massachusetts.

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,* pp. 237-309.

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

John, B. E., Vera, A. H., and Newell, A. (1994). Towards real time GOMS: A model of expert behavior in a highly interactive task. *Behaviour and Information Technology,* **13**, 4, pp. 255-267

John, B. E. (1996) TYPIST: A Theory of Performance In Skilled Typing. *Human-Computer Interaction , 11* (4), pp.321-355.

John, B. E. (1998) Cognitive modeling for Human-Computer Interaction. Invited paper in the *Proceedings of Graphics Interface 98* (Vancouver, British Columbia, Canada, June 18-20, 1998) Canadian Human-Computer Communications Society.

John, B. E. & Altmann, E. M. (1999). The power and constraint provided by an integrative cognitive architecture. Invited paper, Proceedings of the 2nd international conference on cognitive science and the 16th annual meeting of the Japanese Cognitive Science Society joint conference (July 27-30, 1999. Tokyo, Japan). pp. 20-25.

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. & Lallement, Y. (2000) A Demonstration of Integrative Modeling of a Complex Dynamic Computer-based Task. In *Proceedings of the 2000 AAAI Fall Symposium on Simulating Human Agents*, November 3-5, 2000.

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

Kirwan, B. & Ainsworth, L. K. (Eds.) (1992). *A guide to task analysis*. London, UK

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

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press. Cambridge, Massachusetts.

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

Taatgen, N. A. & Lee, F.J. (submitted). Production Composition: A simple theory of Complex Skill Acquisition.