A Demonstration of Integrative Modeling of a Complex Dynamic Computer-based Task

Bonnie E. John
Human-Computer Interaction Institute
and Department of Psychology
Carnegie Mellon University
Pittsburgh PA 15213
bej@cs.cmu.edu

Yannick Lallement
Novator Systems
444 Yonge Street
Toronto, Ontario, M5B 2H4
yannick@novator.com

Abstract

To be effective in many HCI uses, a computational cognitive model must behave like a human, not simply get the job done with the least effort or in the least time. Building computational models that emulate human behavior on complex tasks is an ambitious undertaking, as they must include perception and motor behavior as well as problem-solving, learning and large amounts of knowledge. This project demonstrates that building a complex model of perception, cognition and motor behavior can be accomplished by re-using architectures and models of component capabilities built for other tasks. It also demonstrates how emergent properties of such a complex model can reveal important insights for cognitive science.

Introduction

To be effective in many HCI uses, a computational cognitive model must behave like a human, not simply get the job done with the least effort or in the least time. To that end, it may have to emulate the perceptual, cognitive and/or motor processes people go through to complete a task. It may have to take the same amount of time that people take to perform a task. It may make the same type of errors people make. It may have to take the same amount of time and require the same type of experience to learn to perform a task. It may have to do the same inefficient fumbling for a solution to a difficult problem.

Building computational models that emulate human behavior on complex HCI tasks is an ambitious undertaking. If the models include perceptual, cognitive and motor actions, interesting (and sometimes counter-intuitive) interactions exist because complex tasks require these capabilities to perform in parallel (e.g., Gray, John & Atwood, 1996). If models of real tasks include problem-solving, they require both general problem-solving strategies and heuristics as well as means to utilize domain knowledge. If learning is included with these other aspects, such models are charting new waters in cognitive science research. However, we believe such models are important to cognitive science research and that the state of computational modeling has reached a point at which an attempt can be made at these combinations.

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). We believe model reuse is a profitable avenue to explore for four reasons. First, it make intuitive sense. Consider natural language (NL) comprehension as an example. People learn to comprehend natural language as children and use this skill in every task that requires it throughout their lives. A model of natural language comprehension built to comprehend an extensive corpora of sentences should be applicable to many HCI tasks; a new model of NL comprehension should not be built from scratch for each new task. Second, reuse provides expertise beyond a single researcher. That is, the NL-comprehension model can be built by NL researchers and the complex model-builder benefits from their expertise. Third, reuse provides external verification of the component models. If the NL-comprehension model comprehends the NL corpora well and it comprehends the error messages in a complex task, these two sets of data provides independent tests of the mechanisms of that

model. Finally, reuse provides additional constraint on models of complex tasks. If the NL-comprehension model comprehends the NL corpora well, the HCI modeler cannot change its basic mechanisms simply to make it work in the new domain. Additional vocabulary is likely to be necessary, and perhaps even additional grammatic structure, but not changes to the comprehension mechanism.

We demonstrate the reuse approach by modeling initial learning in the Kanfer-Ackerman Air Traffic Control© (KA-ATC©) task (Ackerman & Kanfer, 1994). We will first present the KA-ATC© task. We will then examine the capabilities necessary to perform that task and present our solutions to modeling these capabilities, be it through reuse of an existing architectural mechanism, previously-built model, or a new part of the model added from scratch. We present the operation of KA-ATC© model and discuss the lessons learned about modeling human behavior and the reuse modeling approach.

The KA-ATC© Task

The KA-ATC© (Ackerman, 1988; Ackerman & Kanfer, 1994) is a dynamic task where participants are presented with the start-up screen shown in Figure 1. Planes are represented by lines of information in the upper left corner of the screen including their flight number, type of aircraft, scheduled landing time, amount of fuel remaining, and position in the hold pattern. These planes is must be moved down to runways in the lower-left corner before they run out of fuel. The planes are moved between adjacent hold-levels and from hold-level 1 to the runways using cursor-movement and function keys. A complex set of rules constrain which planes can land on which runways depending on the wind direction, wind speed, and weather conditions displayed in the upper right corner of the screen.

The participants were trained to perform the task with a self-paced, online system. The training phase was composed of 61 screens presenting the objects involved in the task, explaining the subtasks and procedures to follow to do the task, and making the participant practice those procedures. The three subtasks are: accept a plane from the queue, move a plane inside the hold pattern and land a plane. When the participant starts the task, he or she has already practiced each of the three subtasks once.

Capabilities Necessary to Perform the Task

Performing the KA-ATC© task involves the coordination of many different capabilities. Some of these capabilities are fundamental to interactive behavior, like being able to perceive the world and act on it. Other more complex capabilities are skills learned prior to walking up the to KA-ATC© task, like knowing how to learn a procedure from written instructions. We used the fundamental capabilities to select an architecture, and the more complex capabilities to select previously-existing models to incorporate into our effort to build a model of this complete task

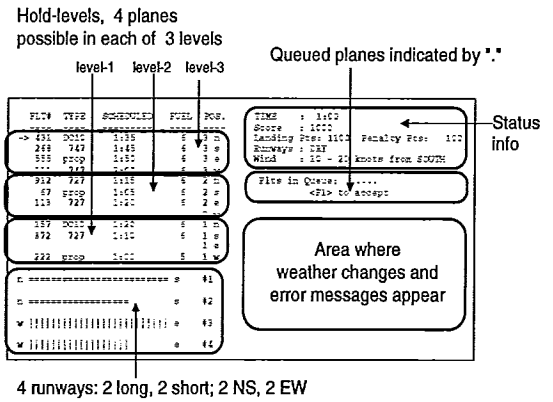

Figure 1. Annotated screen of the KA-ATC© task.

Fundamental capabilities and architecture

Fundamental performance capabilities required to do the KA-ATC© task include

- Visually perceiving objects on the screen (the task is silent, no audition is necessary)
- Pressing keys on the keyboard
- Deciding what key to press based on knowledge of the task and what is perceived on the screen
- Learning the task both from the on-line instructions and improving through practice

These requirements helped us select a cognitive architecture to use. Several candidates fit these requirements (e.g., ACT-RPM, Byrne & Anderson, 1998; EPIC-Soar, Chong & Laird, 1997). We chose EPIC-Soar because of our prior experience writing a model of a dynamic computer-based task (Lallement & John, 1998) and with model reuse in Soar (Nelson, Lehman & John, 1994). EPIC-Soar (Chong & Laird, 1997) is a combination of perceptual and motor processors from EPIC (Meyer & Kieras, 1997) and Soar's cognitive processor (Newell, 1990). We use a modeling idiom called PEACTIDM which allows Soar to work with EPIC without jamming its peripherals (Lallement & John, 1998). Describing these architectures, their interaction, and the PEACTIDM idiom is beyond the scope of this paper, but details can be found in the references cited above.

Complex capabilities and prior models

Beyond simple perception, thinking, motor actions and learning, other slightly more specific capabilities are needed to perform the KA-ATC© task. These capabilities are assumed to be acquired prior to seeing the KA-ATC© task and readily available for a participant to use while performing this task. They include the following.

- Switching attention to react to a dynamic situation
- Learning procedures from instructions
- Learning where objects are likely to be on the screen.

Attention switching. The KA-ATC© task is a dynamic

task where new information appears on the screen every few seconds. Some of this information is crucial to performing the task well. For example, when a star comes up next to a plane ID, it means that the plane has only a few minutes of fuel left and should be landed as soon as possible. Also, when the participant makes an error, a message appears explaining what the error was (e.g., attempting to land a plane in a cross-wind instead of into the wind). Thus, the model must have the capability to interrupt what it is currently doing and pay attention to new information.

The Wickens's task (Martin–Emerson & Wickens, 1992) requires similar attention-switching capabilities. It is a continuous tracking task interrupted by a choice-reaction task. Participants must keep a moving cursor inside a target using a joystick, and respond as fast as possible to a stimulus (left-arrow or right-arrow) appearing on another part of the display by pressing the left button or the right button. Both the Wickens's task and the KA-ATC© task require perception of simple objects, rapid attention switching, decision-making, and keying. We wrote an EPIC-Soar model of the Wickens's task inspired by Chong and Laird (1997) and presented in Lallement and John (1998). That model learns when to propose motor operators. Before learning, the model knows the available motor operators, but not when they are applicable in the real world. Therefore, it generates an impasse at each decision cycle and reasons about operators in an internal "imagined" world to determine which operator to propose. After learning, it knows when each action is applicable and proposes appropriate operators without impasse.

Our model of the KA-ATC© task is built on top of the Wickens's task model, that is, it actually reuses all the Wickens's task code and the KA-ATC© model can perform the Wickens's task. When it performs the KA-ATC© task it uses that part of the Wickens's code applicable to the new task. The re-used code directs the models' attention to new objects appearing on the screen -- - which is important for both tasks. The two models also share the same object identification mechanism, which allows attention to be interrupted, described in Lallement and John (1998). When they issue a saccade to identify a new object on the screen, EPIC simulates the delay of human perception so the features of this object are not immediately available. As soon as the features are available, the object is recognized and verified by separate operators. However, before the features of the objects are available in WM, other cognitive operations can take place if they are applicable. These operations could conceivably direct attention away from this object. Finally, the two models use the PEACTIDM idiom for modeling immediate behavior tasks, and the same Encode and Decode mechanisms.

Learning procedures from instructions. Vera, Lewis and Lerch (1993) employed a standard Soar multi-stage learning idiom applied to learning to perform a given task without instructions, after once performing the task with the instructions. Their model was originally applied to withdrawing money from an automatic teller machine (ATM). This model is particularly relevant to the KA-ATC© task because the instructional paradigm is very similar. In the same way that a person withdrawing money has to follow the successive instructions that appear on the ATM screen in order to reach the goal, participants in the KA-ATC© task read instructions online and are asked to practice the actions described by the instructions.

The ATM model first comprehends the instructions and creates an internal representation called a *behavior model* that represents them. The first time the model performs the task, still with the instructions visible, the behavior model is interpreted, which results in new rules representing operationalized instructions. In order for these rules to fire when the instructions are not present, Soar needs to go into a subspace and guess what actions it can take. The environment prompts the possible actions; in the case of an ATM, the slot will prompt inserting a card, the keyboard will prompt typing a PIN, etc. In that subspace, Soar keeps guessing possible actions, until it guesses the right one (here, "insert the card into the slot"). This results in the model recognizing that this is indeed the correct next step and learning a new rule dependent only on the environmental affordances. The next time the model will execute the task, the environment will directly prompt what actions can be taken, without going into a guessing space.

In the case of the KA-ATC© task, the participant performs the task once during the training session, while he reads the instructions. We did not explicitly model the learning that occurs during this first performance of the task; instead we hand-coded the rules that would be the result of this phase. These rules are of two types:

- rules that encode a behavior model for which subtask to do to accomplish the current task (for example, find-arrow is the first subtask in the task to land a plane)
- rules that encode a behavior model for which action to take to accomplish the current subtask (for example, the look-around action will be useful for the find-arrow subtask; the press-key action will be useful for the move-arrow-to-plane subtask).

These hand-coded rules come into play the first time the model performs the KA-ATC© task. Thus, a full behavior model is encoded, which is analogous to the participant's state after the training phase, when there has been no further practice to improve upon these rules.

Learning where objects are likely to be on the screen. Evidence from eye-movement studies on the KA-ATC© task show that people initially perform a lot more saccades to many more areas of the screen when they start performing the task than they do after substantial practice (Lee, 1999). That is, they learn to move their eyes more efficiently, a skill that occurs in many tasks as well as the KA-ATC© task (e.g., Haider & Frensch, 1999). Therefore, the model must have the capability to learn to move its eyes more efficiently. The model must do this within the

constraints of the Soar architecture and therefore reuses the multi-stage learning idiom introduced earlier.

In particular, when the screen first appears, the model randomly looks around at the objects on the screen. When the model decides to land a plane, it sets itself a subtask to find the arrow. When it finds it, the model uses a memorization process while looking at the arrow. It imagines an image of the screen to generate candidate positions of the arrow and tests these against the current position of its eye (supplied by EPIC). The result of this memorization process allows the model to saccade directly to the upper left corner of the screen on the next trial when it wants to find the arrow.

Knowledge included in ATC-Soar prior to performance. In addition to the mechanisms discussed above, the ATC-Soar model is given knowledge about the KA-ATC© task itself. This knowledge is in the form of hand-written Soar-productions which include knowledge about the goal structure, the layout of the screen, and important aspects of the task.

The model uses an explicit 4-level goal manipulation mechanism. Every goal level is represented explicitly in WM.

1. The activity (Wickens, ATC)
2. The task (in ATC: accept-plane, move-plane, land-plane)
3. The subtask (in land-plane: find-arrow, find-plane, move-arrow-to-plane, select-plane, find-runway, move-arrow-to-runway, land-plane).
4. The action (look-around, press-key).

The activity level determines which task the model will perform. Set by the analyst using the simulation, the model can perform either the Wickens's task or the KA-ATC© task. In the following, we are interested only in the KA-ATC© task performance; the Wickens's performance is described in Lallement and John (1998).

The task and subtask levels are defined in the rules of the KA-ATC© task. The action is the motor level; the model can either press a specific key or look at the screen. The training materials explicitly define the necessary keypresses, and implicitly define visual-search operations. For instance, the first instruction about landing a plane is as follows.

```
Press the <UP ARROW> or <DOWN ARROW>
keys until the arrow on the screen
points to the plane you want to land.
```

From this, we encode knowledge that when the task is to land a plane, and the location of the arrow is unknown, set a subtask to find the arrow. Likewise, when the location of the desired plane is unknown, set a subtask to find a plane to land. When both of these locations are known, set a subtask to move the arrow to the plane, and so on. These pieces of knowledge form the behavior model discussed above.

We encoded knowledge given by the instructions about location of elements of the screen; for example, the runways are located in the lower left portion of the screen. We also gave the model some extra knowledge about the general zones of the screen and the environment, which was not explicitly mentioned in the training session. For example, the model knows that the arrow is in the left part of the screen (because it always in that area in the training sessions); it knows that the information presented on one line correspond to one plane; it knows that the keyboard is under the screen (during the training session, participants are asked to locate the various keys used in the task). While this knowledge does not come from any specific instruction, we assume it came from practicing during the learning phase (position of the arrow, grouping on one line of plane information), or from prior experience (position of the keyboard, that information on a single row is related).

ATC-Soar also begins with some knowledge about important aspects of the task which are described in the instructions. In particular, it knows that when a star appears next to a plane that plane should be landed as soon as possible.

We specifically did not include the rules about which runway to choose in the initial implementation of the model. The instructions did not force practice in applying these rules and human participants often made errors related to these rules. Thus, they did not fully learn the rules from the instructions; either they did not comprehend them, they did not know how to apply them or they could not retrieve them in context. As a result of the absence of these rules in our model, when the it needs to select a runway, it chooses one randomly, enabling the possibility for errors. When the model makes an error, the onset of an error message will trigger more learning phases, but this happens after the first land-plane phase that we cover in this article, so we will not discuss it further.

Qualitative Performance of the Model

When ATC-Soar begins the KA-ATC© task a startup screen appears with four planes in the lowest hold-level. The model looks around its *world* (a simulated screen and keyboard) recognizing objects (e.g., plane numbers, fuel indicators, and specific keys). When a star appears next to a plane, EPIC's eye is automatically drawn to the star because of its sudden onset. ATC-Soar recognizes the star as a signal to land a plane and sets this goal for itself.

At the task level, ATC-Soar knows that landing a plane requires a procedure with many subtasks, and it proposes to do them all. It impasses to choose between these subtasks and uses the mechanism from Vera, et al. (1993), to learn the order for future use (encoded in a *task-chunk*). In this case, the first subtask is to find the cursor-arrow. ATC-Soar then visually searches around the screen, issuing action-level commands to do so, until it finds the arrow. It then uses the memorization mechanism described above to learn where the arrow is for future use (encoded in a *visual-chunk*). This knowledge is particularly useful in the KA-ATC© task because the arrow always appears in the upper left on the start-up screen.

The model continues with the next step in the landing procedure, move-arrow-to-plane. To do this it must learn

the appropriate keypress. It looks at the keyboard to generate candidate keys (up-arrow, down-arrow, enter, or F1). It again uses the Vera, et. al. (1993) mechanism to select and learn these *action-chunks*. Once the arrow is pointing to the proper plane, the model uses the same mechanisms to select that plane by pressing the enter key.

Once the plane is selected the model must choose a runway to land it on. As discussed above, we did not encode the rules for choosing runways by hand, so the model simply chooses one at random. When it attempts to land on that runway, if an error message appears, the model chooses another runway at random and attempts to land the same plane on that runway. (Although not yet implemented, the Vera, et. al. (1993) mechanism could be used to comprehend the error message and actually learn the runway-selection rules.)

In this way the model lands the four planes on its start-up screen. It does a lot of looking around the screen and keyboard, and pondering what to do next. In the course of this performance, it learns where the arrow appears at the start-up of the task (visual-chunks), in which order to do subtasks (task-chunks), and which keys to press when it wants to move a plane from one place to another (action-chunks). The second time it performs the task, this learned knowledge comes into play and makes its performance much more efficient. In fact, the model goes from an average of 25 seconds to land the first plane in the first trial to 19 seconds to land the first plane in the second trial, comparable to the 28 seconds and 16 seconds, respectively, recorded for a typical human participant (i.e., subject 309, Ackerman, 1988).

Preliminary Lessons and Future Work

This paper reflects a preliminary look at this model, its behavior and emerging characteristics. We will first look at the preliminary lessons for learning research and the process of modeling complex tasks. We will then outline the essential work still remaining, to remind the reader to consider the preliminary lessons with an appropriate skepticism.

Preliminary Examination of Learning

One advantage to having a computational cognitive model of a task is that the model can be manipulated in ways in which human participants cannot be manipulated. In particular, we can turn Soar's learning mechanism on or off, whereas it is difficult to prevent a human from learning as they perform a task. We can also examine the relative influence of different types of learning on the performance of this task.

We categorized what ATC-Soar learned as *visual-chunks, task-chunks,* and *action-chunks.* Visual-chunks capture knowledge that helps make visual information gathering in the KA-ATC© task more efficient. Visual-chunks include the knowledge about where the arrow is on the startup screen and also knowledge about when it is profitable to look around the screen when doing this task (e.g. when you don't have anything else to do). 5.2 vision-chunks were learned, on average, in each of 25 runs of the first trial[1]. Task-chunks encode information learned about the order of subtasks. 10 task-chunks were learned in each of 25 runs the first trial. Action-chunks encode when to hit specific keys. 33.9 action-chunks were learned, on average, in each of 25 runs in the first trial.

We separated these chunks by type and selectively loaded them into the model so that we had one version of the model that had "learned" only vision-chunks, one version that had "learned" only task-chunks, and one version that had "learned" only action-chunks. We also had one version using all the chunks. We ran each version of the model 25 times, with Soar's learning mechanism turned off, to achieve the results shown in Figure 2. We ran these models with learning off to determine the relative contribution of each type of learned knowledge in isolation from other possible learned knowledge.

	No-chunks	Vision chunks only	Task chunks only	Action chunks only	All chunks
Avg no. of decision cycles to land the 1st plane on Trial-2					
Cycles	495.4	460.7	463.6	429.8	369.5
Std. Dev.	32.1	35.5	34.9	34.5	29.7
Improvement over No-chunks (in decision cycles)					
		34.8	31.9	65.6	125.9
Improve-ment per chunk (in decision cycles)					
		6.7	3.2	1.9	2.6

Figure 2. Performance of the model with learning off, averaged over 25 runs, when different groups of chunks are loaded.

An ANOVA shows a significant difference ($p<0.01$) between conditions. Pair-wise comparisons show a significant difference between no-chunks and all other conditions at the $p<0.01$ level, and between all-chunks and all other conditions at that same level. There is no significant difference between vision-chunks-only and task-chunks-only, but the differences are significant between vision-chunks-only and action-chunks-only and between task-chunks-only and action-chunks-only ($p<0.01$).

The results in Figure 2 show that each group of chunks contributes considerable improvement to the performance of the task. This implies that for a task like the KA-ATC© task, people will improve the efficiency with which they gather visual information, the efficiency with which they make task-level decisions, and the efficiency with which they select keystroke-level actions, all contributing substantially to their performance in doing the task.

[1] Because the model is so complex, and there is an element of randomization in its behavior as it scans the screen, we ran the model 25 times and present averaged numerical results.

Therefore models of learning which hope to simulate human-scale learning need to account for at least visual, cognitive, and motor learning. Models that ignore any of these aspects are probably missing a good portion of learning behavior.

Another phenomenon prevalent in human behavior, that people often switch high-level strategies while doing complex tasks, is not represented either in the performance data of the first two trials of the KA-ATC© task or in our model. Indeed, this behavior has been observed in the KA-ATC© task and switching to a more efficient strategy significantly improves performance (John & Lallement, 1997). The behavior required to emulate such strategy switching is beyond the scope of our current model, but a complete model of human performance in this and similar tasks would have to include this complexity. However, even where no strategy-switching is evident in human behavior, as in the first two trials of the KA-ATC© task, performance improvement stems from the contributions of learning in different aspects of human performance, including at least perception cognition and motor actions.

Preliminary Lessons for Modeling Complex Tasks

The ATC-Soar project serves as an existence proof that architecture-, mechanism- and model-reuse can be used to create a model of a more complex task from components created for other, usually simpler, tasks. Importantly, the ATC-Soar model does not lose its ability to perform the simpler Wickens's task, that is, the mechanisms and knowledge to perform the more complex task do not interfere with performing the simpler task. Prior research that re-used Soar code (Nelson, et. al., 1994) did uncover interference between assumptions underlying the contributing models, which made their combined behavior cognitively implausible, but we learned from that enterprise and such problems did not occur in this project.

Anecdotally, we, as modelers, found the model-building enterprise more constrained when re-using architectures, mechanisms and code. This constraint is very positive (John & Altmann, 1999) because there are a large number of decisions to make in modeling a complex task. Re-using decisions made to serve other modeling purposes, and having them work well to model our task, gave us confidence that these decisions are not atotally rbitrary, but are converging on a style of modeling that will transfer to many tasks.

Future Work

The lessons discussed above must be viewed in light of the preliminary point of this research. The first essential question not answered in this paper is how this model matches human behavior on this task. Although several of the mechanisms contributing to the model have been validated separately in other domains, the ATC-Soar model must be evaluated as a whole to ensure that these components retain their cognitive plausibility as they are integrated in a more complex model. We are currently examining the performance of the model as a whole in comparison to human performance at a deeper level than the simple performance time reported above.

Assuming the model produces reasonably human-like behavior, the model itself can then be analyzed to discover to what degree each capability contributes to the resulting performance. Only then will we understand the actual costs and benefits of architecture- and model-reuse from a model-building point of view. The next step in integrative modeling research will be to model a new complex task that uses the same capabilities, re-using all the ATC-Soar code, and adding only knowledge of the new domain, rather than additional mechanisms. Given the immaturity of this research, the next effort will almost certainly fail for some aspects of the next task and new mechanisms will be required. The hope is that each additional task will require fewer new mechanisms, until we converge on a set of cognitively plausible mechanisms that are sufficient to model a large number of complex and important tasks.

Acknowledgments. We thank Rick Lewis for his help hand-writing rules which would have resulted from his model for learning by instruction. This research is supported by the ONR, Cognitive Science Program, Contract Number N00014-89-J-1975N158. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ONR or the US Government.

References

Ackerman, P. L. (1988). Determinants of individual differences during skill acquisition: cognitive abilities and information processing. *Journal of experimental psychology: general, 117,* 288–318.

Ackerman, P. L. & Kanfer, R. (1994). *Kanfer-Ackerman air traffic controller task CD-ROM database, data-collection program, and playback program.* Technical report, University of Minnesota, Department of Psychology.

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

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.

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

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,* 237-309.

Haider, H & Frensch, P. A. (1999) Eye movement during skill acquisition: More evidence for the information-reduction hypothesis. *Journal of Experimental*

Psychology: Learning, Memory, and Cogntion, 25, 1, 172-190.

John, B. E. & Altmann, E. M. (1999). The power and constraint provided by an integrative cognitive architecture. *Proceedings of the Second International Conference on Cognitive Science and the 16th Annual Meeting of the Japanese Cognitive Science Society Joint Conference* (July 27-30, 1999. Tokyo, Japan).

John, B. E. & Lallement, Y. (1997). Strategy use while learning to perform the Kanfer-Ackerman air traffic controller task. *Proceedings of the nineteen annual Cognitive Science Society Conference*, Stanford University.

Lallement, Y. & John, B. E. (1998). Cognitive architecture and modeling idiom: an examination of three models of the Wickens's task. *Proceedings of the twentieth annual Cognitive Science Society Conference*, Madison, WI.

Lee, F. J. (1999). *Does learning of a complex task have to be complex? A case study in learning decomposition.* PhD thesis, Carnegie Mellon University.

Martin-Emerson, R. & Wickens, C. D. (1992). The vertical visual field and implications for the head-up display. *Proceedings of the Human Factors Society.*

Meyer, D. E. & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple task performance: Part 1. Basic mechanisms. *Psychological Review, 104*(1), 3–65.

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, 353-358.

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

Vera, A. H., Lewis, R. L., & Lerch, F. J. (1993). Situated decision-making and recognition-based learning: applying symbolic theories to interactive tasks. *Proceedings of the 15th annual conference of the Cognitive Science Society*, Boulder, CO.