Identifying Dual-Task Executive Process Knowledge using EPIC-Soar

Ronald S. Chong and **John E. Laird**

Artificial Intelligence Laboratory
University of Michigan
Ann Arbor, MI 48109-2110
{rchong,laird}@umich.edu

Abstract

In this paper, we present a lineage of models that is used to identify the additional knowledge required to perform two tasks concurrently at an expert level. The underlying architecture used for this modeling is EPIC-Soar, a combination of the sensory and motor modules of EPIC, and the cognitive processing of Soar. Within EPIC-Soar, we build models for the Wickens' task, a combination of tracking and choice-reaction time tasks. A key product of the models is an identification of the knowledge required to combine these two tasks: the executive process knowledge. We also demonstrate that it is possible to learn *some* of this knowledge through experience. We achieve performance comparable, in terms of error-rates and reaction times, to human data and an EPIC model.

Introduction

Possibly the most persuasive reason to study dual-task acquisition and performance is because it can give us insights into the architecture of the mind. Such situations stress human capabilities to the extent that the observed patterns of behaviors set constraints on the human information-processing architecture, sometimes leading to detailed hypotheses about the cognitive architecture (Meyer & Kieras, 1997a, 1997b).

Damos and Wickens (1980) reported a study that demonstrated the existence and effect of timesharing skill on the performance of a dual-task combination. This timesharing skill is often referred to as the "executive process". It is the job of the executive to control or mediate the execution of the tasks that need to be performed concurrently.

From a computational modeling perspective, we define the "executive process" as the knowledge necessary to model dual-task behaviors above and beyond that which is required to do the two tasks individually. Two reasonable questions to ask are, "What is the nature of the knowledge that describes the executive process," and, "How can this knowledge be learned?"

The goal of our work is to begin to answer these difficult questions for the class of simple perceptual-motor tasks. Our approach is to first identify and classify the knowledge of the executive process (which is presented here), and then to develop plausible task-independent learning procedures for acquiring the knowledge. Our early results suggest that some of the knowledge required to orchestrate dual tasks can be learned from failures through experience with knowledge compilation mechanisms such as chunking.

The Wickens' Task

In our work, we use a task combination we call the Wickens' Task (Martin-Emerson & Wickens, 1992). It consists of a tracking and a choice-reaction time task. The task environment as shown in Figure 1 was used in a study of dual-task performance for the purpose of evaluating the effect of vertical separation on the tracking and choice task. The application of this study is to the design of the heads-up displays used in aviation.

The experimental setup included two displays: the tracking window and the information display. The tracking window contained a cursor and a target circle. In the tracking task, the subject used a joystick to keep a cursor (which is always moving) in the target circle. This task simulated a pilot tracking a ground target when landing.

The choice-reaction task stimulus was presented in the information display, where either a left or right arrow would periodically appear. The stimulus duration was one second. When the stimulus appeared, the subject was to press one of two buttons beneath their middle and index fingers on the left hand; left button for the left arrow, right button for the right arrow. This task simulated warnings or other indications requiring some sort of immediate response that may appear while a pilot is landing.

The task requirements were for the subject to keep the cursor in the target, but to respond to the choice stimulus on the information display as soon as possible. RMS tracking error,

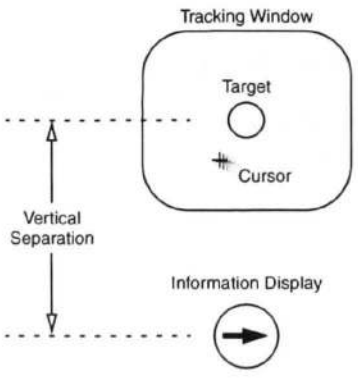

Figure 1: The Wickens' task environment.

reaction time data, and response correctness were gathered. The tracking error was recorded only during the one-second presentation of the choice stimulus and the one-second immediately following the stimulus offset because it was anticipated that tracking errors due to allocation of attention to the choice task would occur during this period.

In the original study, several experimental variables were manipulated, two in particular: the vertical separation between the tracking and choice displays (separations ranged from zero degrees (superimposed) to 35.2 degrees); the difficulty of the tracking task (considered to be either "high" or "low"). For our work, we also varied the vertical separation, but fixed the tracking difficulty at "high". This condition puts the most stress on the model and, as a result, makes the effects of changes to the model most evident.

The Architecture

Our approach to modeling behavior is to start with a set of fixed assumptions that are realized in a computational architecture. For our work we have created a hybrid, which consists of the sensor and motor modules of EPIC and the cognitive module of Soar.

EPIC

EPIC (Executive Process-Interactive Control) (Meyer & Kieras, in press) is an architecture whose primary goal is to account for detailed human dual-task performance. It extends the work begun with the Model Human Processor, MHP (Card, Moran, & Newell, 1983).

Like MHP, EPIC consists of a collection of processor and memories. There are three classes of processors: perceptual, cognitive, and motor. However, EPIC is distinguished from MHP in two very significant ways. First, the EPIC processors and memories are much more elaborate, each representing a synthesis of the most recent empirical evidence describing psychological phenomena. Secondly, EPIC is a system that can be programmed and executed. When performing a task, these processors run asynchronously with one another.

There are three perceptual processors, visual, auditory and tactile, which receive inputs from simulated physical sensors. The output of these processors is sent to the working memory of the cognitive processor. The cognitive processor consists of working memory, long-term memory, and production memory, and a multi-match, multi-fire production system. The cognitive processor has no learning mechanism. On receiving input from the perceptual processors, it performs the reasoning necessary for the task being modeled and sends output commands to the motor processors, of which there are three: ocular, vocal, and manual.

Every EPIC model requires an executive process, encoded as productions, whose purpose is to coordinate the progress of the other processes (tasks) in the model. The executive process does not take part in accomplishing the tasks directly (such as sending motor commands in service of a task); it only modulates the activity of the tasks.

Soar

Soar is a general architecture for building artificially intelligent systems and for modeling human behavior (Rosenbloom, Laird, & Newell, 1993; Newell, 1990). Soar has been used to model central human capabilities such as learning, problem solving, planning, search, natural language and HCI tasks. Soar is a goal-oriented architecture, where the primitive deliberative act consists of the selection and application of operators. Soar's long-term knowledge is encoded as productions, which carry out the basic functions of selecting and applying operators. Soar's sensory information and current situational analysis are held in a declarative working memory, which is matched against production memory. Goals automatically arise when the knowledge encoded as productions is insufficient to directly select or apply an operator. In the subgoals, this basic processing recurs, so that productions in the subgoal will match to select and apply operators in service of determining which operator to select or to apply in the supergoal. Soar incorporates a *single* learning mechanism called chunking which compiles the problem solving in the subgoals into productions. In combination with various problem solving methods, chunking has been found to be sufficient for a wide variety of learning (Lewis, *et al.*, 1990; Miller and Laird, 1996).

EPIC-Soar

EPIC-Soar is an integration of the perceptual and motor processor models of EPIC, and Soar. This is an attempt to get the best of both worlds: the detailed predictions and explanations of sensory and motor systems from EPIC (an ability Soar does not possess), and the broader, cognitive problem solving, planning, and learning capabilities of Soar (an ability EPIC does not possess).

To create the hybrid architecture, we performed a "mind transplant"; EPIC's cognitive processor was replaced with Soar. EPIC and Soar remain independent programs which communicate using socket connections. Soar now accepts EPIC perceptual and motor processor messages as input to its working memory, and returns motor processor commands to EPIC as output. The cycle of interaction and information exchange between the systems is as follows: EPIC sends perceptual and motor messages to Soar and then waits; Soar accepts the inputs, runs for one decision cycle, returns to EPIC any motor commands that may have been generated and then waits; EPIC accepts the motor commands and executes them. This cycle repeats.

EPIC On The Wickens' Task

EPIC has previously been used to produce a quantitatively accurate model for the Wickens' task (Kieras, 1994). In the

EPIC system, the Wickens' Task is a simulation that runs asynchronously to the other components (the perceptual, cognitive, and motor processors) of the EPIC architecture. The perceptual and motor processor interact with the simulation to perceive the world and to effect changes to the world. As can be seen in Figure 2, the EPIC model provides a good match to the empirical data on both the reaction-time and tracking error measures.

Applying EPIC-Soar to the Wickens' Task

The production rules of the EPIC model realize a model of expert dual-task performance. Many of these rules explicitly control the coordination of the two tasks; they are the "executive process". These executive process rules "micro-manage" the execution of the tasks, providing deliberate control to intermix the component actions of the two tasks. Also, EPIC does not attempt to explain why these executive process rules exist or how they are learned.

Our intent is to replicate the EPIC results in EPIC-Soar using a a less task-dependent executive and to explore ideas of how expert executive process knowledge can be learned.

Before modeling the Wickens' Task within EPIC-Soar, we needed to make some changes to both EPIC and Soar. Due to limited space, we will give only a brief discussion of these changes.

First, we found it necessary to add new motor and perceptual processor status messages to EPIC. These messages allowed us to reduce the amount of knowledge needed to perform the task and freed Soar from having to compute and maintain information that is readily available from the processors. These messages were not needed in the EPIC model because of its use of explicit control of the tasks. The EPIC-Soar approach on the other hand is one of minimal control, relying on the state of the processors to guide behavior.

The changes to Soar were made to correct a problem with the post-learning behavior of cognitive models (Wray, Laird, & Chong, 1997). These changes provided an architectural approach to producing the correct behavior and will be included in future releases of Soar.

Identifying Executive Process Knowledge

The rest of this paper will present a lineage of six models (one individual, two sequential, and three concurrent) that were used to transition from novice behavior to expert behavior for the Wickens' Task. Each model in the lineage represents the addition of knowledge to the previous model. We relied heavily on the existing EPIC expert model to guide us in building of our own models.

Models Of The Individual Tasks

We first built models of the *individual* tasks. In Soar, a task is encoded as the operators that must be selected and applied to perform the functions of the task. Some of these operators will be primitive, and will be carried out directly as productions which change internal data structure or send motor commands to EPIC. These operators map almost directly onto the productions in EPIC which take 50 ms of simulated time to execute. Soar productions are finer-grain than EPIC or ACT-R (Anderson, 1993) productions, with multiple productions contributing to the selection and application of an operator. We adopted this approach in Soar so that the knowledge for selecting an action can be learned and modified independent of the knowledge about performing the action. This also makes it possible to have many different situations in which an operator is selected (because multiple productions can suggest the same operator in different situations), and an operator can lead to different or even parallel actions in different situations (because multiple productions can be involved in performing the actions of an operator).

However, not all operators for a task are directly executable, possibly because of insufficient learning. In these cases, the operators automatically become goals which are solved by selecting and applying additional operators. For the Wickens' task, the top-level set of operators include an operator for the tracking task (`tracking-task`), and the choice task (`choice-task`).

tracking-task	**choice-task**
track-target	*recognize-stim*
watch-cursor	*verify-stim*
	find-response
	send-response

Each of these operators has constituent suboperators listed below. We had no independent empirical guidance for choosing this structure, although the lower-level primitives are consistent with some of the primitives used in the EPIC model, and with earlier models in Soar (Wiesmeyer, 1992).

These operators are sufficient to perform each of these tasks independently, using the EPIC perceptual and motor processors. To generate "expert" behavior and knowledge, each task was run independently using Soar's learning mechanism, which built productions (chunks) that allow the tasks to be performed without the subgoals. These expert versions (the original operators plus the chunks) are used for the following studies.

Strategies for Dual-Task Behavior

At least two dual-task strategies can be used to have tasks run simultaneously. The first is to have the tasks run *sequentially* — do tracking, then stop tracking and switch to the choice task when the stimulus has occurred; after the stimulus has been responded to, resume tracking. A second strategy is to run the tasks *concurrently* with their components parts interleaved. This latter strategy seems most consistent with the instruction as reported by Martin-Emerson & Wickens (1992), "Subjects were instructed to execute a response within the one-second stimulus display period and to divide their attention equally between the two tasks". This is also

the strategy Kieras (1994) found necessary to adopt. We explored both strategies.

Sequential Strategy

The sequential strategy arises by selecting first the tracking-task operator, and then interrupting it with the choice-task operator whenever the choice stimulus appears, then resuming tracking after a response is produced. We manually added two rules to the EPIC-Soar model to achieve this strategy: one rule prefers the tracking task when the stimulus is absent; the other rule prefers the choice task when the stimulus is present. Additionally, since the choice task assumes that the eye is already fixated at the location where the stimulus will appear, an extra rule was needed that would move the eye to the choice stimulus when the vertical separation was sufficient to cause detailed perceptual information of the stimulus to be unavailable. This is the most basic sequential strategy. We ran this model for 300 trials per visual separation condition. (All EPIC and EPIC-Soar models presented in this paper were run at this level.) The average performance on each condition is reported. The RT and RMS tracking error results are plotted in Figure 2 and are labeled *Sequential*.

It is possible to improve the RT match to EPIC of this model by adding knowledge that allows the system to anticipate and prepare for the appearance of the stimulus. When a command is sent to EPIC, it passes through the motor processor in two phases, preparation then execution, both of which take time to perform. However, the total elapsed time of a command can be reduced if, at an earlier time, the command has been prepared for. In the Wickens' task, while the tracking task is taking place, we can prepare the eye to look at the location where the choice stimulus will appear. Thus, when the choice stimulus appears, the rule that fixates the eye on the stimulus will be performed in a shorter time. We manually added a production to generate a preparation for eye movement throughout the tracking task. These results are plotted as *Sequential + prepare* in Figure 2. Here we see that preparing for the stimulus improves the RT match to the EPIC results. The tracking error under both conditions is very high.

The poor prediction of tracking error is the most glaring problem with this sequential strategy. However, there is a straightforward explanation: tracking error is measured from the beginning of the choice task until one second after the choice task completes. Since no tracking was done in the sequential model during the choice task, the error rate is expected to be high. The model strongly suggests that it is necessary for tracking to continue during the choice task; i.e. for both tasks to run concurrently.

Concurrent Strategy

Allowing both tasks to run concurrently is easily done by using the operator composition technique used in earlier Soar work by Covrigaru (1992).

When two tasks are allowed to run concurrently, there is the risk of two or more motor commands being simultaneously sent to the same modality. For example, in the Wickens' task, the model may attempt to respond to the choice stimulus and at the same time attempt to move the joystick, both of which use the manual motor system. In EPIC, this condition is called a "jam" and both commands are ignored.

The original EPIC model avoided jams because of the executive process, which orchestrated the intermixing of the component actions of the two tasks so that jams did not arise. In contrast, our general approach is to try to perform both tasks concurrently as much as possible. When jams arise, the system uses a task-independent recovery mechanism to learn to avoid the jams in the future. Thus, the EPIC-Soar model incrementally learns how to intermix the commands based on failures and experience.

In more detail, when EPIC-Soar jams, a subgoal is automatically created. Within the subgoal, domain-independent operators reconstruct the situation that produced the jam. This creates an internal situation model with which the system can deliberately reason about which action it should take. Within this subgoal, the rules that initially caused the jam will refire. This time, the jam-repair mechanism exam-

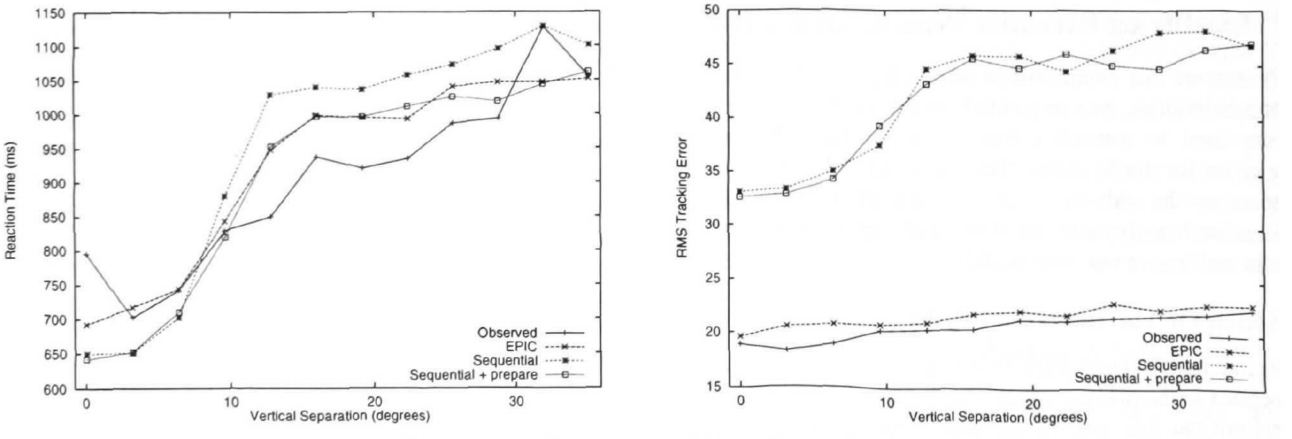

Figure 2: Comparison of the sequential strategy models to observed data and EPIC predictions.

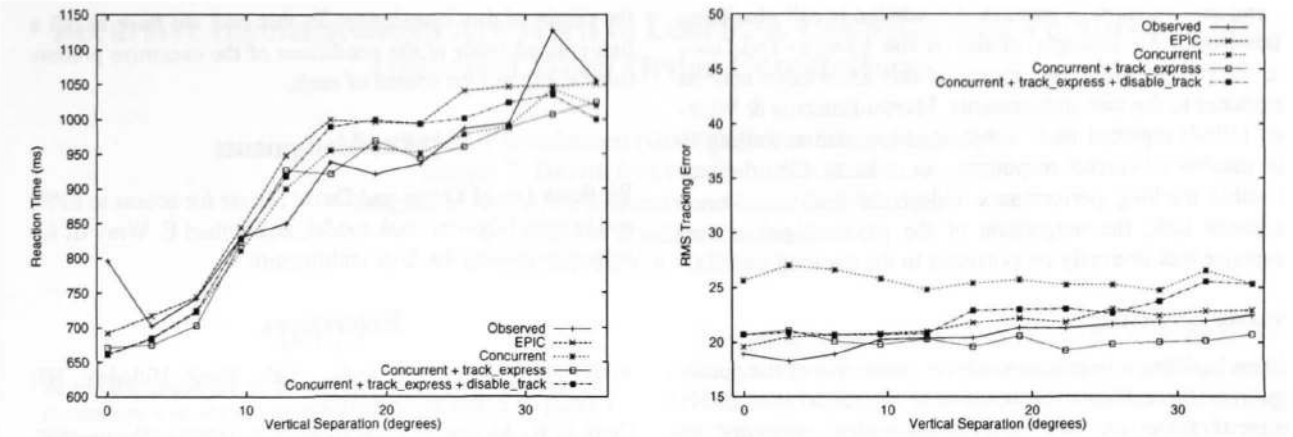

Figure 3: Comparison of the various concurrent strategy models to observed data and EPIC predictions.

ines the motor commands and using the task requirements knowledge (that choice actions are preferred to tracking actions) decides on which command should be sent. A rule is learned so that the next time the state of the world puts the system in the same situation where it could again simultaneously send both commands, this new knowledge "steps in" and produces the preferred command, thus avoiding the jam condition. This mechanism was able to build all the jam avoidance knowledge necessary for this task. The key aspect of this learning is that it transforms the general declarative statement, that choice is to be preferred, into procedural knowledge that applies at the exact point where it is needed, i.e. avoiding a jam. Figure 3 shows the results of this approach labeled *Concurrent*. This model has two qualitative effects: tracking error has been significantly lowered compared to *Sequential + prepare*; tracking error is now independent of increasing vertical separation. This model is still a poor predictor of tracking error.

To improve the model, we returned to the EPIC model and observed that it included a production called `track-follow-it-express`. The purpose of this rule was to track as soon as possible after the choice response had been made. The normal tracking rule fires when the manual motor processor is free. However, during the completion of the choice task, the tracking task can be initiated as soon as the motor processor has finished preparing the choice response. This allows the preparing of the tracking command to overlap with the execution of the choice action. Figure 3 shows the results of this approach labeled *Concurrent + track_express*. With the addition of this rule, the tracking error is now closer to the observed and EPIC data. Interestingly, it predicts better performance; the error is independent of the separation.

To attain the upward slope of the tracking error, we again returned to the EPIC model and found a group of rules which explicitly disabled tracking when the eye was moving from the cursor to the stimulus and to the cursor. The rationale is that no data about the cursor should be available (based on the EPIC visual perception model), so any attempt to track at this time will be in error. To obtain the same behavior, instead of adding an equivalent production as was done before, we instead modified our track rule such that it would fire only if the eye was not moving. We hypothesize that our original task encoding was incorrect, and that the tracking task as originally defined should have this condition. The *Concurrent + track_express + disable_track* traces in Figure 3 show our final results.

Analyzing The Executive Process Knowledge

In this section we review this knowledge and discuss possible mechanisms for its acquisition. Although this is a very simple task, four classes of knowledge can be identified.

Task Requirements Knowledge

Subjects acquire this knowledge from listening to and interpreting task instructions, a complex process involving language comprehension. Previous Soar research (Huffman, 1994; Lewis, Newell, & Polk,1989) has explored the acquisition of procedures from natural language instructions.

Strategic Knowledge

Two forms of strategic knowledge were used: *opportunistic*, and *pipelining*. Opportunistic knowledge appeared in the form of the rule that prepared the eye for movement to the stimulus. Opportunistic prepares are beneficial only for commands where there is some certainty that the action being prepared for will need to be done. In the case of looking at the stimulus, we know that it will always occur and, more importantly, we know where it will always occur. On how this knowledge may be acquired, we hypothesize that since *prepares* are an architectural capability and can be used to increase performance, then there may be a task-independent learning procedure that creates prepare rules based on task knowledge or observed regularities in the task environment.

The second form of strategic knowledge is call *pipelining* knowledge. An example of this is the `track-follow-it-express` rule. The source of this knowledge may be attributed to the task inducements. Martin-Emerson & Wickens (1992) reported that "subjects...were paid according to the number of correct responses;...in order to elicit the best possible tracking performance independent of the discrete response task, the magnitude of the payment per correct response was inversely proportional to the tracking error."

Innate Knowledge

When building a cognitive model in Soar, one of the guidelines used to estimate the veracity of the model is to determine if there are any components (rules, operators, or mechanisms) in the model that could not be learned by the architecture. If this is the case, then the model makes the claim that such components are *innate* meaning that the knowledge exists *a priori* to performing the task and therefore is not learned during performance. The jam-repair mechanism, like other mechanisms such as task instruction acquisition, may be an example of innate knowledge.

Experiential Knowledge

This knowledge is acquired from experience in performing the task. One instance of this knowledge is the rules that were generated by the jam-repair mechanism. Strategic knowledge could also be classed as experiential knowledge though it depends on when the knowledge is acquired.

Summary

In this paper, we presented a lineage of six models that was used to identify the additional knowledge required to perform two tasks concurrently at an expert level for the Wickens' Task. Each model in the lineage represents the addition of knowledge to the previous model.

Most of the knowledge has been manually added and are essentially Soar analogues of the EPIC productions, which on its own is uninteresting. However, our larger goal is to have a system that learns these rules. The merit of this work then is that it takes the first small steps towards that goal: (1) we have a hybrid learning and performance architecture; (2) we have a expert performance model that is now situated in a learning architecture; (3) we can identify the kinds of knowledge needed to progress from novice to expert; (4) we can now posit general task-independent learning procedures to acquire this knowledge; and (5) we have demonstrated one such task-independent acquisition procedure that learns how to deal with some of the initial problems of concurrent performance.

We are continuing to refine our model. When satisfied with its performance, we will pursue a task-independent learning mechanism to acquire experiential strategic knowledge. We have yet to settle on a satisfactory hypothesis for the origin of this knowledge. To this end, we have begun a fine-grained study of the conditions of the executive process rules to explain the source of each.

Acknowledgments

We thank David Kieras and David Meyer for access to EPIC and to their Wickens' task model, and Robert E. Wray III for help in extending the Soar architecture.

References

Anderson, J. R. (1993). *Rules of the Mind.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Covrigaru, A. (1994). Emergence of Meta-Level Control in Multi-Tasking Autonomous Agents. Ph.D. thesis, The University of Michigan, Ann Arbor.

Kieras, D.E. (1994). An introduction to the EPIC architecture for computational models of human performance, In *Proceedings of the Fourteenth Soar Workshop.*

Huffman, S.B. (1994). Instructable Autonomous Agents. Ph.D. thesis, The University of Michigan, Ann Arbor.

Lewis, R.L., Huffman, S.B., John, B.E., Laird, J.E, Lehman, J. F., Newell, A., Rosenbloom, P. S., Simon, T. & Tessler, S. G. (1990). Soar as a unified theory of cognition: Spring 1990. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society.*

Lewis, R.L., Newell, A., & Polk, T.A. (1989). Toward a Soar theory of taking instructions for immediate reasoning tasks. In Proceedings of the Annual Conference of the Cognitive Science Society.

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

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

Meyer, D.E. & Kieras, D.E. (1997b, in press). EPIC: A computational theory of executive cognitive processes and multiple-task performance: Part 2. Psychological Review.

Miller, C.S. & Laird, J.E.(1996). Accounting for Graded Performance within a Discrete Search Framework. *Cognitive Science*, 20(4), 499-537.

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

Rosenbloom, P.S., Laird, J.E., & Newell, A. (1993). *The Soar Papers.* Cambridge, MA: The MIT Press.

Wiesmeyer, M. (1992). An Operator-Based Model of Covert Visual Attention. Ph.D. thesis, The University of Michigan, Ann Arbor.

Wray, R.E., Laird, J.E., & Chong, R.S. (1997). Improving Timing Predictions Following Operator Compilation in Soar. Unpublished report.