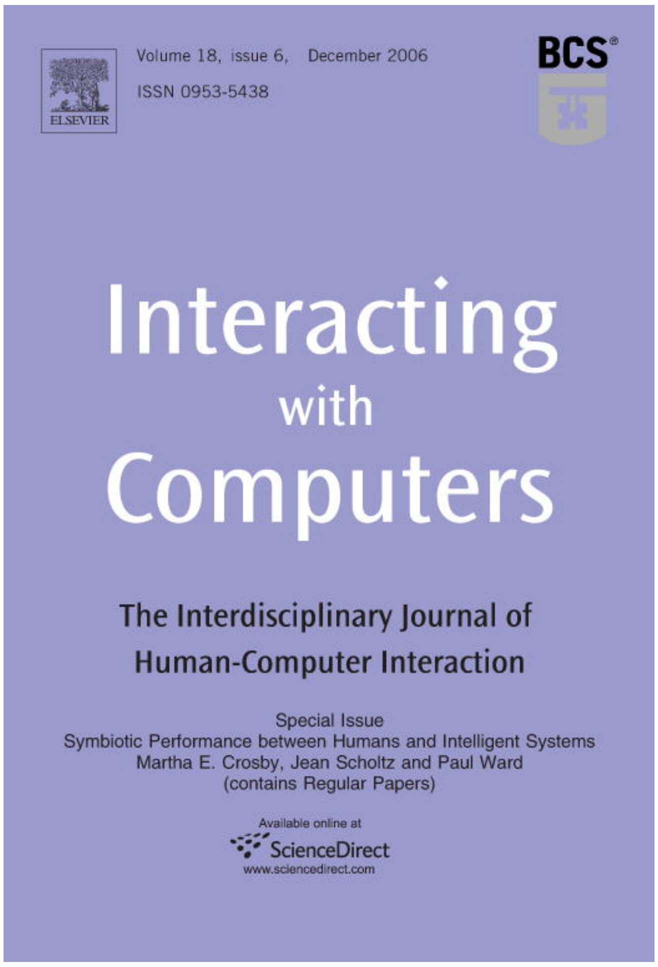

Volume 18, issue 6, December 2006
ISSN 0953-5438
ELSEVIER
BCS
Interacting with Computers
The Interdisciplinary Journal of Human-Computer Interaction
Special Issue
Symbiotic Performance between Humans and Intelligent Systems
Martha E. Crosby, Jean Scholtz and Paul Ward
(contains Regular Papers)
Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com
ScienceDirect
Interacting with Computers 18 (2006) 1215–1241
Interacting with Computers
www.elsevier.com/locate/intcom

Toward a real-time model-based training system ☆

Wai-Tat Fu [a,*], Daniel Bothell [b], Scott Douglass [b], Craig Haimson [c], Myeong-Ho Sohn [d], John Anderson [b]

[a] *Human Factors Division and Beckman Institute, University of Illinois at Urbana-Champaign, One Airport Road, Savoy, IL 61874, USA*
[b] *Carnegie Mellon University, Pittsburgh, PA 15213, USA*
[c] *Aptima, Inc, Washington, DC 20005, USA*
[d] *George Washington University, DC 20052, USA*

Available online 11 September 2006

Abstract

This article describes the development of a real-time model-based training system that provides adaptive "over-the-shoulder" (OTS) instructions to trainees as they learn to perform an Anti-Air Warfare Coordinator (AAWC) task. The long-term goal is to develop a system that will provide real-time instructional materials based on learners' actions, so that eventually the initial set of instructions on a task can be strengthened, complemented, or overridden at different stages of training. The training system is based on the ACT-R architecture, which serves as the theoretical background for the cognitive model that monitors the learning process of the trainee. An experiment was designed to study the impact of OTS instructions on learning. Results showed that while OTS instructions facilitated short-term learning, (a) they took time away from the processing of current information, (b) their effects tended to decay rapidly in initial stages of training, and (c) their effects on training diminished when the OTS instructions were proceduralized in later stages of training. A cognitive model that learned from both the

☆ This research is supported by the ONR Grant N00014-99-10097. Part of the data in this article was reported in a conference paper in the International Conference on Cognitive Modeling, 2004. The current article is substantially expanded from the conference paper. We thank Paul Ward and three anonymous reviewers for their thoughtful comments and suggestions on this article.

* Corresponding author. Tel.: +1 217 244 8617.
E-mail address: wfu@uiuc.edu (W.-T. Fu).

0953-5438/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.intcom.2006.07.011

upfront and OTS instructions was created and provided good fits to the learning and performance data collected from human participants. Our results suggest that to fully capture the symbiotic performance between humans and intelligent training systems, it is important to closely monitor the learning process of the trainee so that instructional interventions can be delivered effectively at different stages of training. We proposed that such a flexible system can be developed based on an adaptive cognitive model that provides real-time predictions on learning and performance.

Keywords: Model-based training system; Cognitive model; Skill acquisition; Training; ACT-R; Model tracing; Augmented cognition

1. Introduction

There has been a long history of research in cognitive psychology that studies human skill acquisition in a variety of contexts. Unfortunately, there is still a gap between laboratory research in cognitive psychology and many real-world concerns on training, such as how to derive the most effective set of instructions, how to facilitate knowledge acquisition in various stages of training, or how to maintain the same level of performance over time in different problems or situations. As Newell (1973) lamented, research in cognitive psychology has failed to provide a characterization that integrates various components in human cognition. Such integration is essential for many real-world applications. Newell proposed the concept of a cognitive architecture as a solution to the issue of integration. Cognitive architectures are computational systems that integrate different aspects of the human cognitive system, such as perception, attention, and memory, so that cognitive models can be constructed based on the architecture to produce coherent human behavior in different tasks. Cognitive architectures can also be used as a theoretical basis for the development of intelligent training systems. The idea is that training materials should be designed with reference to a cognitive model of competence that the trainee is being asked to learn. This means that the cognitive model should keep track of the learning process and the various cognitive states in real time, and inform the training system to deliver training material in ways that facilitate the effectiveness of training. The coupling of cognitive models and intelligent training systems can therefore greatly enhance the symbiotic interactions between humans and intelligent systems.

This article focuses on the development of a computerized model-based training system based on the ACT-R 5.0 architecture (Anderson et al., 2004). As a first step towards this goal, we will describe an experiment and a model of how people learned from "over-the-shoulder" instructions as they played the role of an Anti-Air Warfare Coordinator (AAWC). The goals of the experiment are to show that the impact of these over-the-shoulder (OTS) instructions varies at different stages of training in a highly interactive and dynamic problem-solving task, and to provide data to test the validity of the model that captures the learning process. The long-term goal is to develop a system that will provide real-time adaptive feedback on participants' actions, so that eventually the initial set of instructions on a task can be strengthened, complemented, or overridden at different stages of training. To preview our results, we found that OTS instructions had different short-term and long-term effects on training effectiveness at different stages of training. We concluded that to fully capture the symbiotic performance between humans and intelligent training systems, it is important to have a flexible system that adapts to the learning process of the trainee, so that OTS instructions can be effectively delivered to maximize their effects on training.

In the next section, we will first provide a brief review of model-based training systems and various cognitive theories behind these systems. We will then describe the ACT-R architecture that serves as the theoretical basis of our system. We then provide a description of the current status of the real-time model-based training system, discuss the details of the AAWC task, and how OTS instructions are generated from the system for this task. Finally, we present details of an experiment that tested the impact of OTS instruction at different stages of learning, and the implications on the design of systems that attempt to capture, assess, and evaluate skilled and symbiotic performance between humans and intelligent training systems.

2. Model-based training systems

There has been a long history of applying cognitive theory of learning and skill acquisition to model-based training systems (e.g., Anderson et al., 1995; Graesser et al., 2004; Hill and Johnson, 1993; Sleeman and Brown, 1982). The key idea of a model-based training system is that instructions should be given based on a cognitive model of the competence that the trainee is being asked to learn. In other words, the cognitive model should incorporate the underlying skills that allow the model to perform the task the trainee is expected to perform. Based on the model, the system can monitor actions of the trainee and infer the intentions of the trainee by mapping actions of the trainee to components of the model. In other words, a model of competence provides an explanation of actions as trainees interact with the system. Immediate feedback or real-time instructions can then be given to the trainee to facilitate learning.

Many cognitive theories have been implemented as computational cognitive models to predict performance during learning and skill acquisition. One of the most studied theories of skill acquisition is the ACT* theory, which has been applied to training systems in Lisp programming, algebra, and geometry (see Anderson et al., 1995). Another cognitive architecture that is actively used by many researchers is SOAR (Laird et al., 1987). SOAR provides an integrated problem solving and learning architecture. Tasks in SOAR involve goal-oriented search through a hierarchy of problem spaces. Learning in SOAR occurs when the results returned from a subgoal are converted into new productions that can be applied to achieve the same results under similar circumstances. These learned productions are called chunks, and they summarize both the goal and state context in which an operator applies. Chunks are added to the production memory as they are learned, and constitute the recognition knowledge in the architecture. In many ways, SOAR is similar to ACT* and ACT-R, in which productions are used as units of cognitive skills. Attempts have been made to implement SOAR into a model-based training system (e.g., see Hill and Johnson, 1993).

Finally, Kintsch's (1988) construction-integration (CI) theory has also been implemented as models of planning and skill acquisition (e.g., Doane et al., 2000). Specifically, Kintsch's theory presumes that low-level associations between task instructions and existing long-term skills are constructed and used to constrain knowledge activation via a constraint-based integration process. The resulting pattern of context-sensitive knowledge activation is referred to as a situation model and represents the current state of comprehension, which guides the selection of action. The CI theory of comprehension has been successful in explaining a wide range of phenomena, and has shown that comprehension is one of the essential processes in skill acquisition.

In this article, our focus is on applying the ACT-R theory of skill acquisition to monitor performance as people interact with a model-based training system. Instead of modeling how people comprehend task instructions and construct action plans as in CI models, most ACT-R models assume that instructions are first encoded as propositions and stored as a set of declarative elements. The ACT-R theory has specific predictions on how these declarative instructions are strengthened by repeated exposures, decay with time, and are transformed into procedural skills gradually as these instructions are retrieved and executed. The main assumption in the ACT-R theory is that a cognitive skill consists of units of goal-related knowledge. Cognitive skill acquisition involves the formulation of thousands of rules relating task goals and task states to actions and consequences. Trainee actions are mapped to the set of rules by a process called model tracing (described below), which allows the system to provide the necessary instruction or feedback to the trainee.

A major challenge in building the model is to understand how people respond to and learn from immediate feedback on their actions from the training system, and how it influences the existing declarative and procedural knowledge, and thus the progress of learning. This problem is by no means the only problem that needs work for a realistic real-time model-based training system, but we intend to use it to demonstrate the overall framework and our progress toward developing such a system. To study this problem, we designed an experiment and developed a model that learned from real-time over-the-shoulder instructions in a dynamic task. Before we begin describing the system, we will first briefly describe the ACT 5.0 architecture. Our focus will be on the terminologies and mechanisms that are most relevant to the description of the training system.

3. The ACT-R architecture

One of the basic assumptions in the ACT-R theory is that human cognition emerges through an interaction between a procedural memory and a declarative memory system. In recent development, in addition to the two memory systems, the basic architecture of ACT-R 5.0 consists of a set of modules, each devoted to processing a different kind of information (see Fig. 1). Coordination in the behavior of these modules is achieved through the central production system. A production is a condition-action pair, and at any particular production cycle, one production will be selected and the action will be executed. A production is selected when its condition side matches the current activities in the modules, but the central production system is not sensitive to most of the activity of these modules but rather can only respond to a limited amount of information that is deposited in the buffers of these modules (i.e., the "ACT-R buffers" in Fig. 1). The core production system recognizes patterns in these buffers by matching them to the condition sides of the productions stored in the procedural memory system, and may make changes to these buffers through the action side of the productions – as for instance, when it makes a request to perform an action in the motor module, or a request to move attention in the visual field in the visual module. However, once requests are sent to different modules, activities inside each module can occur in parallel. For example, a single production can send a request to the visual modules to move attention and another request to the declarative memory system to retrieve some information. The latencies of moving attention in the visual modules and retrieving an item from declarative memory are independent, and thus the processes in each module are sometimes asynchronous. As will be described later, this mixture of serial and parallel processing is essential for the model presented in this paper. However, we will first describe the major mechanisms of the ACT-R 5.0 architecture that are most relevant for the model in this article.

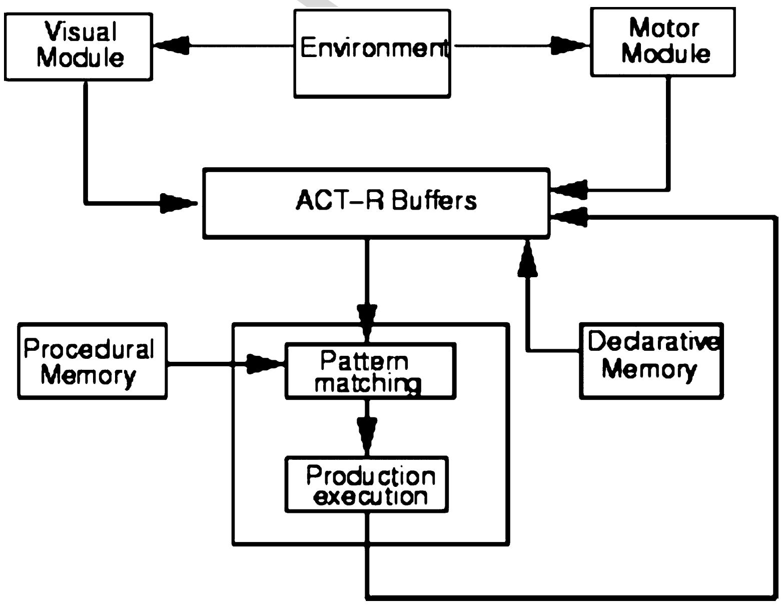


Fig. 1. The ACT-R 5.0 architecture (adapted from http://act-r.psy.cmu.edu/about/). ACT-R has two major memory systems (the procedural and declarative memory systems) and a set of modules (e.g., visual, motor, aural, etc.). Coordination of behavior in these modules is conducted through the central production system (by pattern matching and production execution) and the ACT-R buffers.

3.1. Declarative memory

Declarative learning results in the acquisition of various facts such as the fact that $3 + 4 = 7$. Access to information in declarative memory is controlled by an activation process. The activation of an element in declarative memory (a chunk) is a sum of base-level activation, reflecting its general usefulness in the past, and an associative activation, reflecting its relevance to the current context. The activation of a chunk i (A_i) is defined as

$$A_i = B_i + \sum_j W_j S_{ji} + \varepsilon \qquad \text{(activation equation)}$$

where B_i is the base-level activation of the chunk i, the W_j reflect the attentional weighting of the elements that are part of the current goal, S_{ji} are the strengths of association from the elements j to chunk i, and ε is activation noise sampled from a logistic distribution. The base-level activation rises and falls with practice and delay according to the equation

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) \qquad \text{(base-level learning equation)}$$

where t_j is the time since the jth practice of an item. The equation implies that each presentation of an item has an impact on its activation that decays away as a power function. In the ACT-R community, .5 has emerged as the default value of the parameter d over a large range of applications.

Chunks will be retrieved only if their activation is over a threshold. Because activation values are noisy, there is only a certain probability that any chunk will be above threshold. The probability that the activation will be greater than a threshold x is given by the following equation:

$$P_i = \frac{1}{1 + \exp((\tau - A_i)/s)} \qquad \text{(probability of retrieval equation)}$$

where s controls the value of ε in the activation equation, and is typically set at about 0.4. If a chunk is successfully retrieved, the latency of retrieval will reflect the activation of a chunk. The time to retrieve the chunk is given as

$$T_i = F\mathrm{e}^{-Ai} \qquad \text{(latency of retrieval equation)}$$

where F is a constant that is often set to 1.

3.2. Visual module

The perceptual-motor system of ACT-R 5.0 is based on the EPIC system by Meyers and Kieras (1997), but there are some major differences. For example, in the ACT-R visual system, vision is separated into two modules, each with an associated buffer: A visual-location (*where*) and a visual-object (*what*) module. When a production makes a request of the *where* system, the production specifies a series

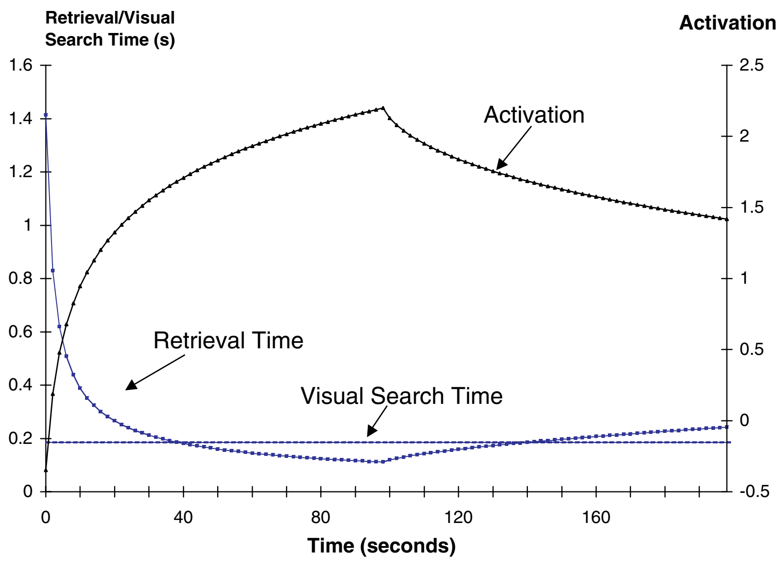


Fig. 2. The activation of a chunk rehearsed every 2 s and its retrieval time. The visual search time is fixed at 0.185 s. The rehearsal stopped at 100 s.

of constraints (e.g., a red circle), and the *where* system returns a chunk representing a location meeting those constraints (a location of the red circle). If there are multiple objects satisfying a request to the *where* system, the location of one will be determined at random. To find the target object may require a self-terminating search through the objects satisfying the constraints. To identify an object, the visual system must send another request to the *what* system. The *what* system will be sent a chunk representing a visual location, with which it can use to shift visual attention to that location. After visual attention is shifted to the object, the *what* system will process the object and returns a chunk representing the object. As an approximation, the time to shift visual attention is fixed at 185 ms regardless of distance[1].

It will be interesting to compare the time to retrieve a chunk and the time to search for information in the world visually. For example, it would be interesting to know whether it will be faster to retrieve some information from memory or to look for the information in the outside world (Fu and Gray, 2000; Gray and Fu, 2004). Fig. 2 shows such a situation, in which the visual search time is fixed at 185 ms, and at the same time the same memory item is being encoded and the activation strengthened according to the base-level learning equation, and how the retrieval time decreases as the activation of the chunk increases (see the latency of retrieval equation). One can see that the retrieval time decreases as the same chunk is encountered every 2 s. After 50 encounters (i.e., after 100 s), strengthening of the chunk stops (i.e., the information is taken away). We can see that after this point, the activation of the chunk decays with time (the rate of decay is controlled by the value of *d*, which is set at 0.5), and the retrieval time increases. We can see that in this

[1] ACT-R assumes that it takes 85 ms to encode the features of a visual object. However, it often requires the firing of two productions, one for finding the location of the visual object, the second for initiating the visual encoding. Each production takes 50 ms. Therefore the total time for a visual search is 185 ms.

example, as the number of previous encounters is more than 20 (i.e., after 40 s of regular encounters), the retrieval time is faster than the visual search time. However, when the encounter stops, the retrieval time eventually becomes larger than the visual search time. The dynamic interactions between the accessibility of "information-in-the-head" and "information-in-the-world" have significant implications in the model presented later.

3.3. Procedural memory

The procedural memory system basically consists of a set of production rules, which detect patterns in the buffers of different modules. At any point in time, multiple productions rules might apply, but because of the seriality in production rule execution, only one can be selected, and this is the one with the highest utility. Utility values are noisy, and the utility of a production *i* is calculated as:

$$U_i = P_i G - C_i + \varepsilon \qquad \text{(production utility equation)}$$

Where P_i is an estimate of the probability that if production *i* is chosen the current goal will be accomplished, *G* is the value of that current goal, C_i is an estimate of the cost to accomplish the goal, and *s* is a random noise sampled from a logistic distribution. The probability that a production will be selected and executed is calculated by

$$P_i = \frac{\exp(U_i/t)}{\sum_j^n \exp(U_j/t)} \qquad \text{(production choice equation)}$$

where the summation is over all applicable productions and *t* controls the noise (ε) in the utilities, which is typically set to 0.5.

Another form of procedural learning is called production compilation, which will take each successive pair of productions and build a single production that has the effect of both. The process bears some similarity to the chunking mechanism in SOAR (Newell, 1990) and is basically a combination of composition and proceduralization in Anderson (1983). Production compilation can be illustrated with respect to a simple paired-associate task. Suppose the following pair of production rules fire in succession to produce recall of a paired associate:

IF reading the word for a paired-associate test and a word is being attended,
THEN retrieve the associate of the word.

IF recalling for a paired-associate test and an associate has been retrieved with response *N*,
THEN type *N*.

These production rules might apply, for instance, when the stimulus *vanilla* is presented: A participant recalls the paired associate *vanilla-7* and produces 7 as an answer. Production compilation collapses these two productions into one. To deal with the fact the second production rule requires the retrieval requested by the first, the product of the retrieval is built into the new production. Thus, ACT-R learns the following production rule:

IF reading the word for a paired-associate test and *vanilla* is being attended, THEN type 7.

This example shows how production rules can be acquired that embed knowledge from declarative memory. This process speeds up the execution of actions and requires less access to declarative memory.

After a production New is composed from productions Old1 and Old2, whenever New can apply Old1 can also apply. The choice between New, Old1, and whatever other productions might apply will be determined by their utilities. However, the new production New has no prior experience and so its initial probabilities and costs will be determined by the Bayesian priors. We will describe how the prior θ is set for P, noting a similar process applies for C. When New is first created, θ is set to be 0. Thus, there is no chance that the production will be selected. However, whenever it is recreated its θ value is incremented according to the Rescorla–Wagner (Rescorla and Wagner, 1972) or delta rule: $\Delta\theta = a(P-\theta)$ where P is the probability of Old1. This gradual conversion process is also related to the reinforcement-learning mechanism that has shown to be successful in characterizing the procedural learning process in the basal ganglia (Sutton and Barto, 1981; Fu and Anderson, 2006). With practice, if the production rule New is repeatedly created its priori θ will converge on P for the parent Old1. The same will happen for its cost and it will be eventually tried over its parent. If it is actually superior (the typical situation is that the new production has the same P but lower C) it will come to dominate its parent. While our experience with this production rule learning mechanism is relatively limited it seems that a working value of the learning rate a is .05. This mechanism allows new instructions to be proceduralized and replace old instructions; however, the replacement process is slow and is governed by the delta rule. We will show later that this has implications on how frequently real-time OTS instructions should be given to create and strengthen new productions that correspond to the OTS instructions.

4. The ACT-R model-based training system

4.1. Model tracing

One of the major challenges for model-based training systems is how to infer the knowledge states of the trainees by interpreting their actions. This is relatively straightforward in ACT-R, as knowledge states are represented as productions and chunks, and changes in knowledge states can be represented as execution of a sequence of productions. When a trainee interacts with the system, each action executed will be monitored and used to update the knowledge states of the model to align with those of the trainee. This is done through a process called model tracing.

The model-tracing technique keeps track of trainee actions and figures out the appropriate productions that lead to the actions, so that appropriate instructional interventions can be given whenever necessary (see Anderson et al., 1995). In our system, trainee actions will be continually monitored and interpreted by searching for sequences of productions that produce the same behavior. Trainee behavior is compared to the model behavior in the model tracing process, and whenever the trainee behavior deviates from the model, the process is able to "work backwards" through the sequence of productions to figure out what leads to the deviation. For example, if the deviation is due to failure to recognize a piece of crucial information on the screen, the training system can highlight the information and make the trainee aware of their lapse.

To be successful, model-based training systems must be carefully engineered to include production sets sufficient to model all trainee actions within a specific interface and task. Since a trainee's actions are often difficult to predict during their interactions with the system, model tracing often needs to accommodate a large number of alternative courses of action in a space of recognizable action sequences. Many methods have been proposed to simplify this process (e.g., Ritter and Koedinger, 1997). In our system, a program called PLASTIC (Douglass, 2004) was used to automatically recognize the sequence of actions that matches different productions. The PLASTIC program uses a grammar-based pattern matching method, which significantly reduces the complexities of action sequences (e.g., see Fu, 2001). The idea is that, instead of using a large number of productions to monitor user actions, the system uses a grammar-based "template" that recognizes user actions. In the current system, PLASTIC was used to simplify the amount of computations involved in recognizing user actions and deriving appropriate OTS instructions. Despite the different internal representations of user actions, the basic technique of model tracing is the same as that in previous ACT-R-based training systems used in Lisp programming, algebra, or geometry.

4.2. Representation of instructions

The ACT-R theory assumes that goal-independent declarative knowledge initially enters the training system in a form that can be encoded more or less directly from instruction or observation. The acquisition of cognitive skill involves the conversion of this declarative knowledge into production rules through the production compilation process described above. The effect of practice on the accessibility of declarative knowledge can be predicted by the set of equations that govern the activation of chunks, and that on the acquisition of task-specific production rules can be predicted by the equations governing the selection of productions as well as the production compilation process.

In most cognitive models, assumptions need to be made about the structure of the model. In ACT-R models, this takes the form of assuming what chunks and productions are available. This creates an additional degree of freedom for the modeler. Anderson et al. (2004) attempted to eliminate this degree of freedom by introducing a system that takes the initial set of instructions and converts the set of instructions into a declarative representation in the model. The set of instructions are then interpreted by a generic set of production rules. Through the production compilation mechanism, a set of specific productions that directly perform the task are generated as the task is practiced. Although this system can only take instructions in a restricted format, it does provide a way for the model to configure itself based on a given set of instructions (see Appendix A for the differences between the instructions given to the subjects and the instructions parsed by the model).

To study the impact of real-time OTS instructions on learning, we designed an experiment using a real-time dynamic task and measured performance through a 2-day period of training. OTS instructions were given using the model tracing process described above. Based on the ACT-R theory of skill acquisition, the model encodes OTS instructions in real time into the declarative memory system. The OTS instructions are interpreted by the same set of productions that interpret the upfront instructions. In other words, the model processes the upfront and OTS instructions the same way, but as we will show later, the introduction of OTS instructions has some interesting changes to behavior that make them distinct from that of upfront instructions.

4.3. Toward a real-time model-based training system

Most model-based training systems involve a static model of competence of the user, and thus lack the flexibility to adapt to the dynamic learning process of the user. Specifically, most competence models do not take into account the impact of real-time OTS instructions, and therefore fail to effectively predict when and how OTS instructions should be presented to facilitate skill acquisition. Fig. 3 shows the overall structure of the proposed real-time model-based training system. User actions are monitored by the system through the model tracing process, and real-time OTS instructions are given at the appropriate time to facilitate learning. The main idea is to have a flexible ACT-R model that adapts to user behavior during the training process by predicting the impact of the OTS instructions on behavior. To preview our results, our experiment show that OTS instructions are more useful during the early stage of learning where the instructions will facilitate the proceduralization process, but they often slow down performance as they often divert attention away from the main task. Our results highlight the importance to have a flexible model that predicts when these OTS instructions should be given to facilitate the long-term training effectiveness.

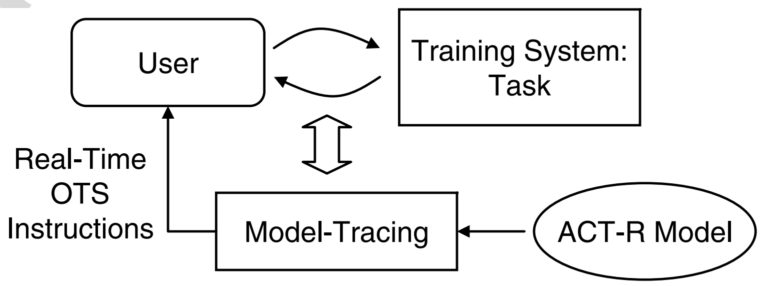


Fig. 3. The overall structure of the real-time model-based system. OTS, over-the-shoulder.

5. The task: the anti-air warfare coordinator

The current task was constructed based on the Georgia Tech Aegis Simulation Program (GT-ASP, Hodge et al., 1995). GT-ASP is a tactical decision-making computer game that simulates tasks facing an anti-air warfare coordinator (AAWC) on board a US Navy cruisers and destroyers. In our task, a participant assumes the role of an AAWC, which includes monitoring a radar screen for unknown aircraft, requesting and collecting information regarding the unknown aircraft, and updating the identity of the aircraft. The task we used is a simplified version of the GT-ASP (we refer to it as the CMU-ASP task henceforth), and a cognitive model was constructed to predict learning behavior and monitor participants' actions though the model tracing process.

The radar screen of the CMU-ASP task (Fig. 4) consists of three major areas. First, the radarscope shows various air tracks. Vectors emanating from the aircraft indicate speed and course. The AAWC is on a ship at the center of the radarscope (called ANZIO). The AAWC moves the mouse within the scope and "hooks" a target airplane by clicking the mouse button. This hooking is necessary whenever the AAWC tries to update identity of unknown aircraft. Second, there is a group of

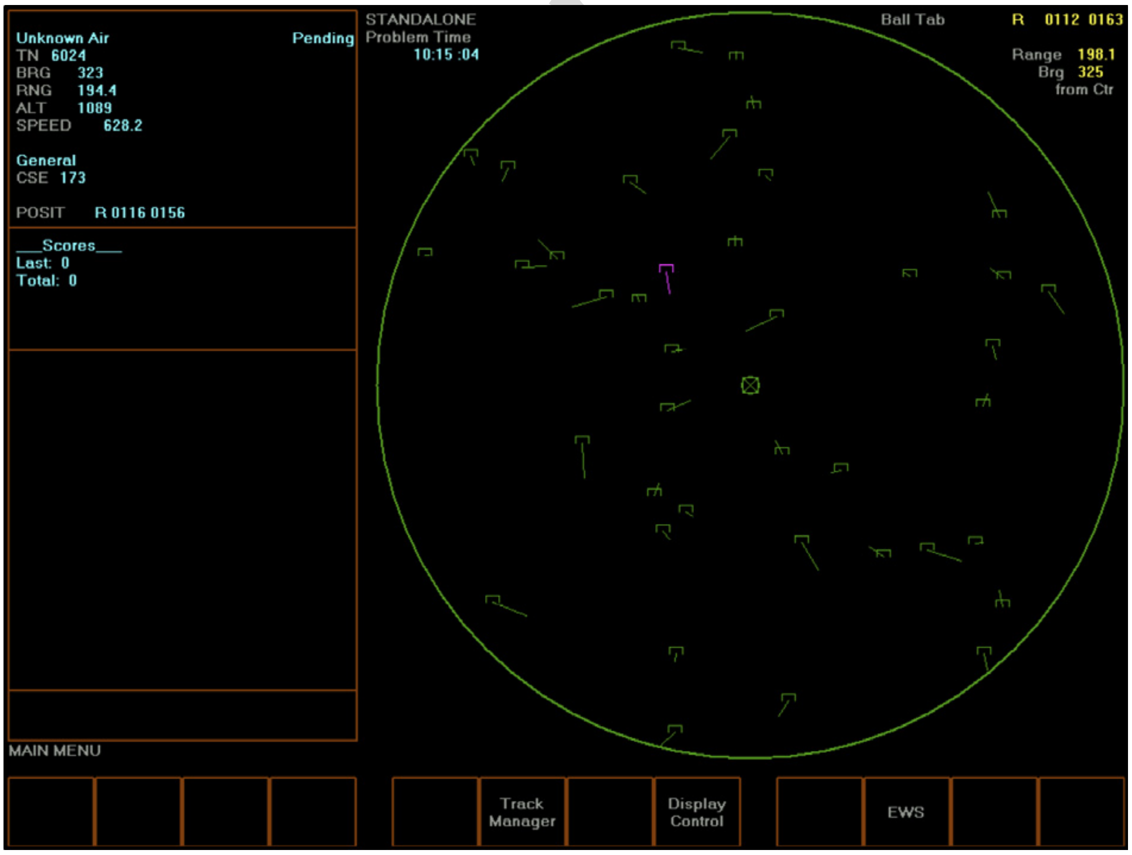

Unknown Air Pending
TN 6024
BRG 323
RNG 194.4
ALT 1089
SPEED 628.2
General
CSE 173
POSIT R 0116 0156
Scores
Last: 0
Total: 0
STANDALONE
Problem Time
10:15 :04
Ball Tab
R 0112 0163
Range 198.1
Brg 325
from Ctr

Fig. 4. The display in the CMU-ASP task.

information boxes on the left of the screen where the participant can get information on tracks. Third, the menu panel shows the current bindings of the function keys (F1 to F12 on the computer keyboard) that are used to issue commands. The AAWC spends the majority of the time in identifying the intent (friendly or hostile) of tracks on the screen and their air type (e.g., strike, or commercial). It is this identification task that the experiment focuses on. We will focus on the first unit task in the identification task – how participants select the next track – with and without the over-the-shoulder instructions. This unit task involves a selection and a visual search component, both of which are subject to significant improvement with the OTS instructions. We will elaborate on this aspect below.

5.1. Instructions

Before the experiment participants were asked to memorize a set of initial instructions (see Appendix A) that were sufficient to finish the task. Half of the participants were assigned to a group where OTS instructions were given (instruction group) and the other half did not receive any OTS instructions (no-instruction group). The OTS instructions were given right after the participants finished a unit task (e.g., after hooking a track or after identifying a track as a commercial profile, an example will be given later). We will focus on the OTS instructions that were given right after participants had hooked a track.

Participants were told that their scores depended on the importance of the tracks classified, but the initial set of instructions did not explain how the importance of a track could be evaluated. After a track had been identified, a score would be displayed in the middle information box on the left of the screen as shown in Fig. 4. The score was calculated by the following final score equation (not known to the participants):

$$\text{Identification score} = (\text{speed} + 3 \times (512 - \text{range})) \times \text{at-me}. \qquad \text{(Final Score Equation)}$$

Hence, the score was high if the speed was high, if it was close to the center of the screen (range was small), or if it was flying towards the ship. The at-me value ranges from 1.0 to 3.0, and it depends on the difference between the course of the track and the bearing of the track measured from the ANZIO. A small difference implies that the track is flying towards the ship, and the at-me value will be high. Since the score decreases linearly over time (from 100% to 25% during the 6-min trial) and there is often insufficient time to classify all tracks in 6 min, the final scores (calculated as the sum of all time-weighted identification scores) depend critically on whether participants can classify tracks in the order of importance.

At any point in time, the identification scores of all unidentified tracks were calculated and tracks were ranked according to their scores. In the instruction group, after the participant hooked a track, if the hooked track was less important than 20% of all unidentified tracks the system would be triggered to give an audio OTS instruction. The most important track on the screen would be highlighted, and instruction would be given to inform the participant why the highlighted track was more important (e.g., "This track was fast and directly approaching the ANZIO"). Participants could then learn from the instructions and improve future performance by selecting the most important unidentified tracks.

6. The experiment

In the process of developing a model-based training system, we conducted an experiment to investigate the impact of OTS instructions to performance at different stages of learning. Thirty-two participants were recruited for a 2-day experiment. Half the participants were assigned to the instruction group and the other half to the no-instruction group. On the first day they were given the set of upfront instructions that taught them how to hook and identify a track, and how to use the F-keys to input the classifications of the tracks. On the first day participants were asked to memorize the set of 13 upfront instructions and were given a paper test afterwards. In the paper test, subjects were asked to fill in the crucial information in the instructions (see Appendix A), and the order of the instructions was randomized. When subjects failed to fill in any of the crucial information, they were asked to study the instructions again, and they were given another test afterwards. Subjects were required to correctly fill in all the instructions and therefore had memorized all the upfront instructions before they began. In other words, during the task, learning was predominately reflected by to extent to which these instructions were proceduralized (i.e., transforming knowledge from declarative to procedural form). The upfront instructions were therefore assumed to be strongly encoded declaratively when subjects began.

Subjects were given 10 6-min scenarios. On the second day they were tested on 10 more. Each of the 20 scenarios was constructed by randomly placing 40 tracks on the screen. Twenty of these tracks satisfied a commercial profile and 22 gave EWS signals and could be classified on those bases (see instructions in Appendix A). These two sets intersected such that 12 could be classified on either basis, and eight tracks that could not be classified on either basis. The 20 scenarios were randomly presented to each subject, but the actual order was recorded so that the same order can be presented to the model.

In the instruction group, OTS instructions were given except the first two and the last two scenarios given on each of the two days. In other words, in the instruction group, participants were given two no-instruction trials, followed by six instruction trials, then two more no-instruction trials, on each of the two days of experiment. OTS instructions were given aurally through the computer speakers.

7. The model

The model uses the same set of parameters as described in Anderson et al. (2004), and uses the same declarative-to-procedural system to process the initial set of

Table 1
The number of participants reported that they used the criteria to select the next track in each group

	Instruction	No-instruction
At-me	15	4
Range	12	10
Speed	15	4

instructions. The current extension processes OTS instructions in a similar fashion, i.e., instructions are initially represented as declarative knowledge and are retrieved when needed. The OTS instructions are also subject to the same declarative-to-procedure conversion as the initial set of instructions. However, because subjects were asked to memorize all the upfront instructions and they all passed the recall test, the OTS instructions are assumed to be weakly encoded compared to the upfront instructions. To preview our results, we did find that the learning of the OTS instructions were slower. Besides, when the OTS instructions were absent, performance of the subjects declined quickly, suggesting that the strength of the OTS instructions decayed quickly before they were proceduralized.

To derive the possible track selection strategies in the model, after the task we asked participants to freely write down the criteria they used to select the next track. The top three criteria were shown in Table 1. When subjects reported that they used the direction of the track as a criterion, they were put under the at-me group; when they reported that they used the distance from the ANZIO as a criterion, they were put in the range group; when the reported that they used the length of the vector or how fast the track was flying as a criterion, they were put in the speed group. Two independent coders conducted the grouping and there was no disagreement between the coders. There were four more criteria reported (e.g., whether the track was closer to the top or bottom of the screen), but none of them was reported by more than two participants. We considered these criteria more random than systematic and have decided not to include them in the model. It shows that participants were able to learn to attend to the right features in the instruction group. It is interesting that even with no instruction, participants tended to choose tracks that were closer to the ANZIO (range), and apparently some of them learned (presumably from the scores they received after identifying each track) that at-me and speed were also important features.

We represented each of the above criteria as a "search-factor" and they were created as declarative chunks when the first over-the-shoulder instruction was given. The search-factors provided constraints to the visual search process. For example, a "fast" search-factor commands a search for a track that has a long vector emanating from the track, and a "range" search-factor commands a search for a track that was close to the ANZIO, and so on.

When the task began, since no search-factor was available, the model started to select the track closest to the ANZIO. This assumption[2] was based on the apparent

[2] This is obviously a post-hoc assumption. Ideally one would have a model that predicts behavior before data are collected, but apparently at this stage we cannot (see also footnote 3).

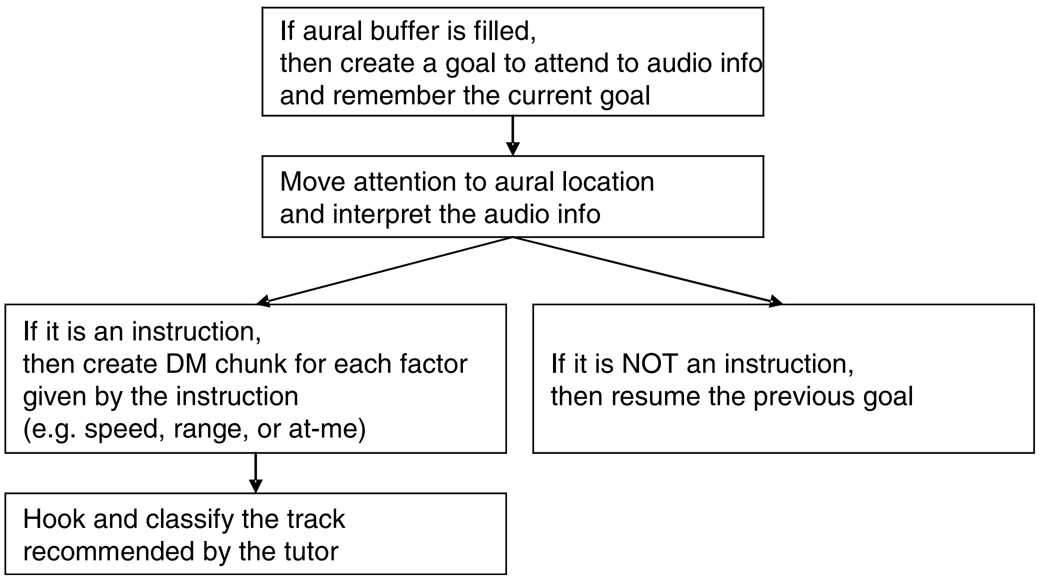

If aural buffer is filled, then create a goal to attend to audio info and remember the current goal
Move attention to aural location and interpret the audio info
If it is an instruction, then create DM chunk for each factor given by the instruction (e.g. speed, range, or at-me)
If it is NOT an instruction, then resume the previous goal
Hook and classify the track recommended by the tutor

Fig. 5. How the model responds to the over-the-shoulder instruction through the aural buffer. (DM, declarative memory.)

tendency for participants to select tracks closer to the ANZIO in both groups (see Table 1). Another assumption was that participants would tend to minimize the search time required to select a track, so that they could classify as many tracks as possible in each trial. Under this assumption, when no additional information is available, the best strategy is to select the track closest to the track that has just been classified, i.e., a nearest-neighbor strategy. This will minimize both the visual search time and the mouse movement time[3].

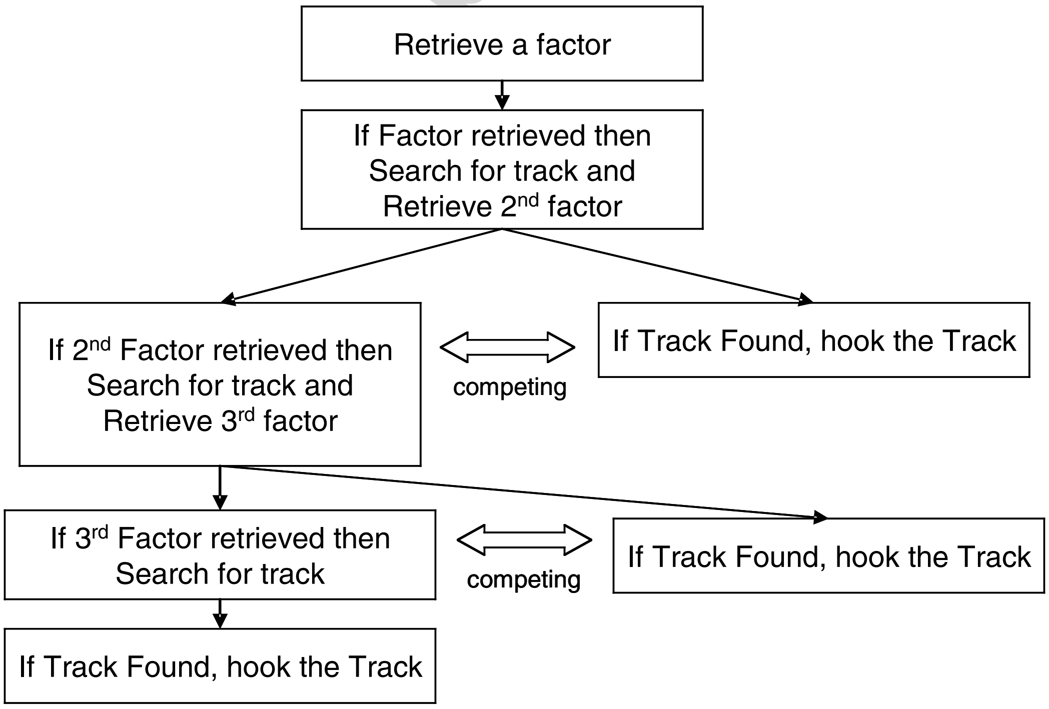
In order for the model to respond to the OTS instructions, we need to provide a mechanism so that goal-directed activities (i.e., production firings) in the model can be interrupted by stimuli in the outside world. We adopted a “buffer-stuffing” approach to this problem. Specifically, when an OTS instruction is given, the instruction will be stored automatically in the buffer of the aural module (which is in parallel to and occurs independently of the central production execution mechanism). At each production cycle (execution of a single production), a production will check whether the buffer of the aural module is filled, and if so, the production will fire (see Fig. 5). This production will also create a new goal of moving attention to this audio signal, and stores the current goal as part of the new goal so that in case the model chooses not to follow the instruction, it can resume the current goal. After attention is moved to the audio signal, the content of the audio signal can be interpreted. If the audio signal is an instruction given by the training system, the model will interpret the instruction (we chose not to model the cognitive complexities involved in the natural language processes involved in the interpretation; instead, the interpretation is done by a lisp function). If the audio signal is not an instruction (for example, it

[3] We have attempted to eliminate these assumptions by having the model randomly picked a track to identify, but the model performed much worse than subjects in the initial trials.

could be informing the radar signal returned from the aircraft), the model will resume the previous goal and activities. This "attention switching" mechanism will slow down performance. In fact, others have successfully showed how this buffer-stuffing mechanism can explain the interruption cost during task switching (e.g., Gray and Schoelles, 2003).

When an instruction is detected, the interpretation process generates search-factors that explain why the track highlighted is important. There are three search-factors given by the over-the-shoulder instruction: speed, range, and at-me. For example, if the instruction is: "This fast track is coming at the ANZIO and needs attention", then the interpretation process will generate the "fast" and "at-me" search-factors. For each of the search-factors generated, the model will create separate chunks representing them in declarative memory. If there is already an identical chunk in declarative memory when the new chunk is created, the chunks will be merged and the activation of the chunk will be increased and its likelihood of being retrieved in the future increases.

After the model repeatedly processes the over-the-shoulder instructions, the activation of the search-factor chunks will be strengthened and the chunks will eventually be retrieved. These search-factors affect the selection of the next track through repeated cycles of retrieval and visual search (see Fig. 2). When the model starts to select a track, it will first try to retrieve a search-factor. If a search-factor, for example, speed, is retrieved, the model will search for a fast track on the radar screen. At the same time, retrieval for the second search-factor is initiated. The visual search and the retrieval compete against each other. If the visual search process finds a track before retrieval finishes, the model will satisfy on the track and stop search-


Retrieve a factor
If Factor retrieved then Search for track and Retrieve 2nd factor
If 2nd Factor retrieved then Search for track and Retrieve 3rd factor
competing
If Track Found, hook the Track
If 3rd Factor retrieved then Search for track
competing
If Track Found, hook the Track
If Track Found, hook the Track
Fig. 6. The model selects the next track retrieving factors that guides the visual search.

ing. However, if retrieval finishes first, a new visual search will be initiated based on the two search-factors retrieved. For example, if the first search-factor is speed and the second factor is range, then the model will search for a track that is both fast and close to the ANZIO. At the same time, retrieval for the third search-factor will be initiated. If, again, a track is found based on the two search-factors before the retrieval for the third search-factor finishes, the model will proceed to hook and identify the track. If retrieval finishes first, a new search will be initiated based on the three search-factors. At this point, since all factors have been retrieved, the model will proceed to hook the track when a track is found. The flow diagram in Fig. 6 therefore shows how top-down influence of instructions can be incrementally combined with the visual search process. The competition of the retrieval and visual search process allows the model to incrementally improve the selection of tracks. The production compilation process will also compile the search-factors into different specific production. For example, if the "speed" search factor is repeatedly retrieved and used, the specific production for searching for a fast track will be created the first time and strengthened subsequently.

8. Results

Fig. 7 shows the final scores obtained after each scenario by the participants and the model in both the instruction and no-instruction groups. The final scores are the total scores obtained for all tracks identified during each 6-min scenario. The difference between the two groups were not significant ($t(15) = 0.95$, $p = 0.36$). The model was able to fit the data well ($R^2 = 0.92$). Participants improved steadily across trials. The final scores dropped slightly at scenario 11, when the participants came back on

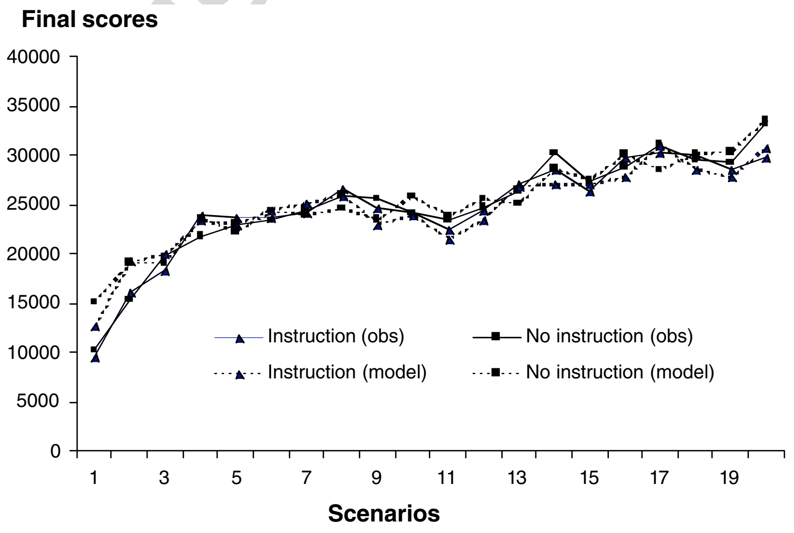


Fig. 7. Final scores obtained by the participants and the model after each scenario. Note that the second day starts on scenario 11. (obs, observed data from participants.)

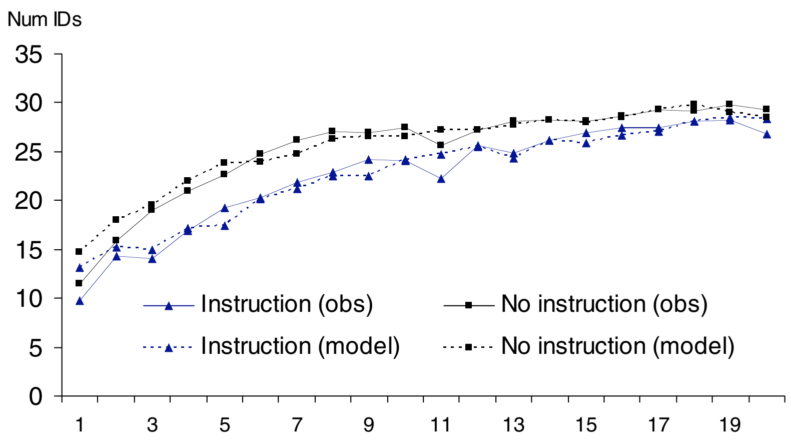


Fig. 8. The number of identifications made in each scenario. (IDs, identifications.)

the second day. In general, the improvements of final scores of the model were similar to those of the participants.

Fig. 7 shows that there was no significant difference in total scores between the instruction and no-instruction groups. However, Fig. 8 shows that the number of identifications made in each scenario was actually higher for the no-instruction group than the instruction group ($t(15) = 3.64$, $p < 0.01$). However, the difference was significant only on the first day ($t(15) = 5.06$, $p < 0.001$), but not on the second day ($t(15) = 0.73$, $p = 0.36$). Apparently the over-the-shoulder instructions had slowed down the participants during the first day. This is most obvious in scenario 3, where the over-the-instructions were first given. Combining the results from Figs. 7 and 8, participants in the no-instruction group had lower scores per each track they identified. In other words, participants in the instruction group had identified more important tracks than the no-instruction group. It suggests that although participants (and the model) were slower performing the task, they were able to choose more important tracks and obtained comparable final scores as participants in the no-instruction group.

Participants sped up steadily across scenarios, except in scenario 11 when participants came back on the second day. The difference between the instruction and no-instruction group also became smaller with practice. It seemed that participants in the no-instruction group reached asymptotic performance earlier than the instruction group. From the analyses in Anderson et al. (2004), during the early stages of learning, time to identify a track depended mostly on cognitive components. At later stage of learning, the time to identify a track depended mostly on perceptual-motor components. The difference between the two groups indicates that the over-the-shoulder instructions may require more cognitive operations to process the instructions. The smaller difference at later stage of learning indicates that the processing of instructions had also sped up. The model was able to capture the patterns of data well in both the instruction and no-instruction groups ($R^2 = 0.96$). The speed-up was captured by the production compilation process in the model, in which both upfront and OTS instructions were slowly compiled into productions with experience.

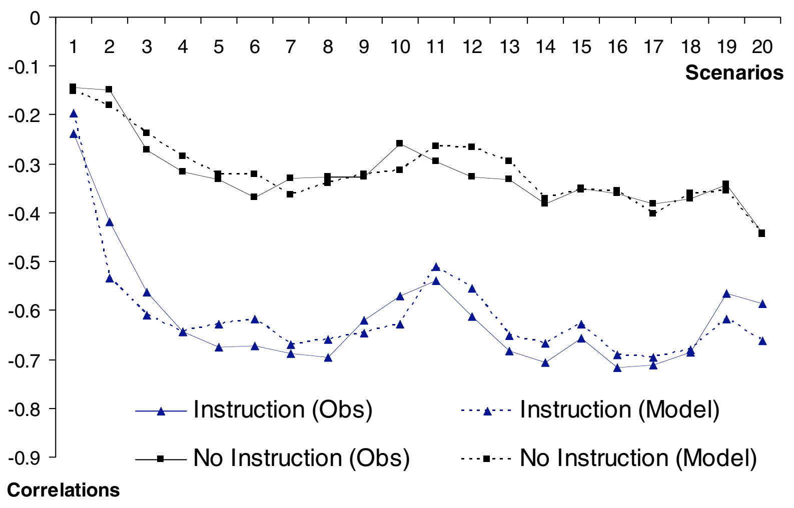


Fig. 9. The correlations between scores of the track identified and their order of identification. The more negative the correlation, the better the identification was in a decreasing order of scores.

To further investigate the effect of the OTS instructions, we derived a measure of how well participants were able to identify the tracks in the order of their importance (i.e., identify the most important first and least important track last). In each scenario, we calculated the Pearson Rank correlation between the scores of the tracks and the order of their identification. For example, if tracks A, B, and C are identified one after the other and the scores obtained are 900, 800, and 600, respectively, then the correlation between the two arrays (900, 800, 600) and (1, 2, 3) is calculated to be −0.98. However, if the tracks were identified in the order of A, C, and B, then the correlation between the two arrays (900, 600, 800) and (1, 2, 3) will be −0.33[4]. In other words, the more negative the correlation, the better the participant is able to identify the tracks in the order of their importance[5].

Fig. 9 shows the Peason Rank correlations across the scenarios for both groups. The correlations for the instruction group were lower than the no-instruction group throughout the 20 scenarios. One may wonder whether the fewer number of identifications may have contributed to the lower correlations (more negative) in the instruction group. However, after checking the data we found that the differences were too big to be caused by the fewer number of identifications (except perhaps in the first two scenarios). In fact, the difference in the number of tracks identified between the two groups was not significant (see Fig. 8) in the second day, the difference in the correlation measure remained large.

The effect of instructions can also be assessed from Fig. 9 at scenarios 9–12, and scenarios 19 and 20 for the instruction group, during which OTS instructions were

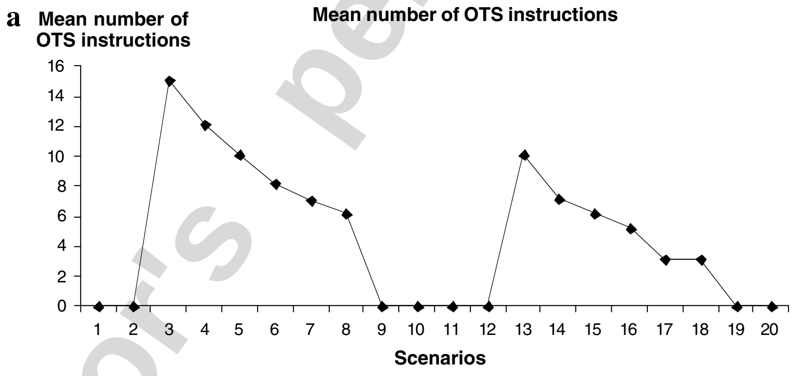
[4] In the actual task, the scores will not be identical when tracks were identified in different orders because scores decrease linearly with time.

[5] The correlation between an interval and an ordinal scale may not reliably reflect the relationship between the two variables if they are highly skewed. However, in our case, the skewness is limited by the design of the task: the scores of a single track were limited to a range from 100 to 1000 and the increments seldom fell below 30.

not given. The instruction group shows some reduction in the magnitude of their correlations but still show stronger (more negative) correlations than the uninstructed group. The model captures the data well ($R^2 = 0.95$). The model exhibited the same decreases of correlations (less negative) when the OTS instructions were absent as shown by the human data. This shows that the model was able to respond and learn from the OTS instructions as the participants.

The nearest-neighbor strategy also did a good job capturing the small learning curve in the no-instruction group. Since the strategy always started with the track closest to the ANZIO, it captured the general tendency to classify tracks closer to the ANZIO first as reported by the participants. Since tracks close to the ANZIO tended to have a high importance, the general tendency therefore led to the sharp decrease of correlation during early trials as the number of identifications increased. However, after tracks close to the ANZIO were identified, the order of identification became more random than systematic, therefore the correlation asymptote at a much lower level (less negative) than that of the instruction group.

To further investigate the impact of each of the search-factors (at-me, range, and speed) across scenarios, we analyzed each of the OTS instructions given and counted the number of times each of the search-factors was emphasized in these instructions. Fig. 10 (a) shows the mean number of OTS instructions and Fig. 10 (b) shows the percentages of the search-factors emphasized in the OTS instructions when subjects performed the task. We can see that, consistent with the correlation measures shown in Fig. 9, the number of OTS instructions decreased across scenarios, indicating that



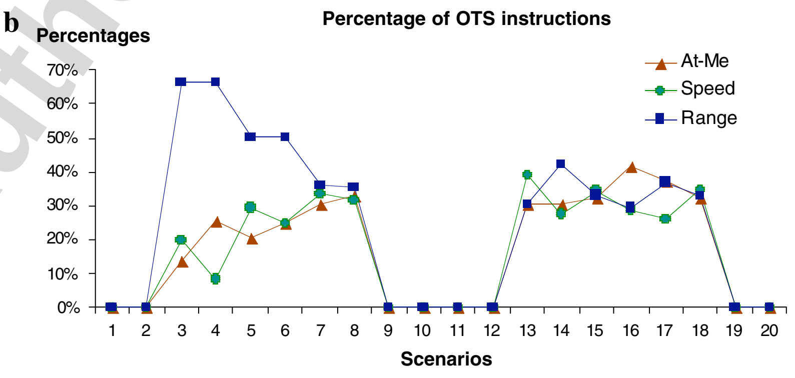


Fig. 10. (a) The mean number of OTS instructions and (b) the mean percentage of each search-factor in the OTS instructions across the 20 scenarios.

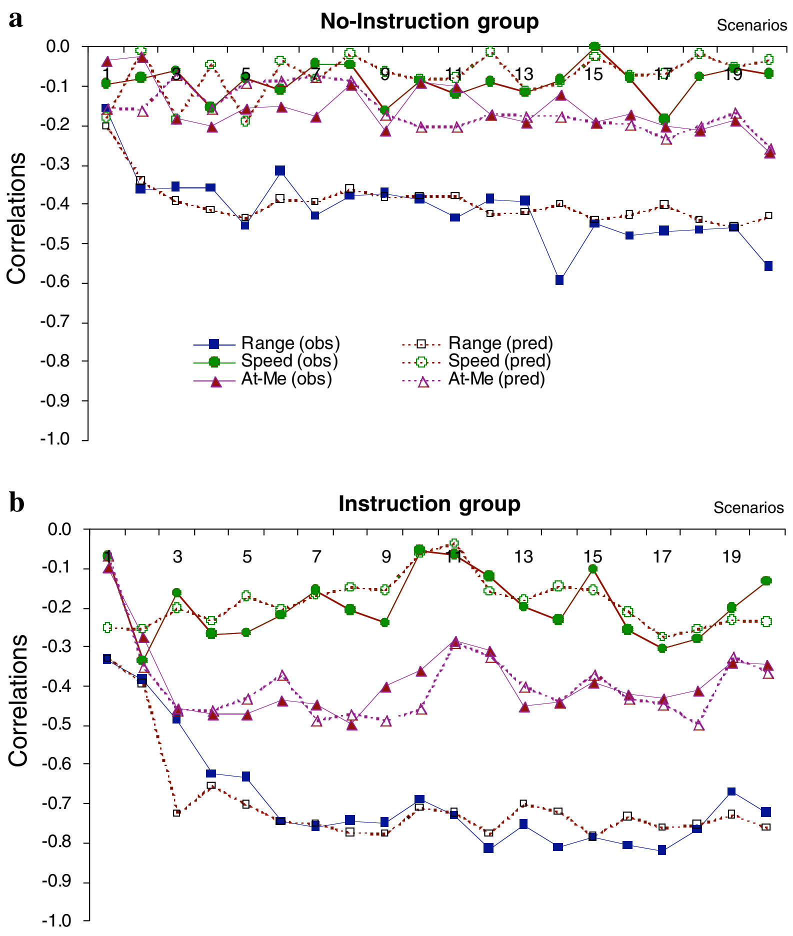


Fig. 11. The correlations between the values of the track in each of the search-factors and its order of identifications across the 20 scenarios in the (a) no-instruction and (b) instruction group. The range values were multiplied by −1 so that they can be compared to the other search-factors. Obs, observed from participants; pred, predicted by the model.

participants became increasingly better at prioritizing the order of identifications (as an OTS instruction was given only when a track that was less important than 20% of all unidentified tracks was hooked). Fig. 10 (b) shows that on the first day, most of the OTS instructions were on the range factor, suggesting that participants paid less attention to important tracks closer to the ANZIO. However, the percentage of OTS instructions that emphasized the range factor decreased while the other two factors increased towards the end of the first day. On the second day, the percentages of all three factors were roughly the same. The trend showed how the OTS instructions impact the choice of the next track: subjects learned to adapt to the weight of the three search-factors as defined by the final score equation through repeated OTS instructions.

Fig. 11 shows the correlations between the values of each of the search-factors of the track identified with its order of identification[6]. In other words, it is similar to

[6] For example, if subjects began the task and identified track A, B, and C. Then the orders of identification for tracks A, B, and C are 1, 2, and 3, respectively.

Fig. 9 except that we used the values of each of the search factor instead of the combined scores. For both the no-instruction and instruction groups, participants paid most attention to range throughout the 20 scenarios, followed by at-me and Speed[7]. However, the correlations were much stronger for the instruction group for all three search-factors. The high percentage of the range factor in the OTS instructions (see Fig. 10) had provided the strongest impact on behavior, as shown by the strong correlations in Fig. 11 (b). The impact on range was maintained even in scenarios when OTS instructions were not available However, for at-me and Speed, the correlations were much weaker during scenarios 9 through 12, and scenarios 19 and 20, suggesting that the impact on behavior was relatively weak: participants paid attention to these two factors when OTS instructions were given, but not when the OTS instructions were taken away.

After inspecting the model, we found that since there were more OTS instructions on range initially, the model had sufficient experiences using range as a search-factor to look for the next track, and the resulting new production created by the production compilation process had a high enough utility value to be selected in later scenarios. Since the utility of the new production does not decay with time[8], the range search-factor was used consistently even in scenarios when the OTS instructions were not available. On the other hand, we found that the model did not have enough experiences to fully proceduralize the at-me and Speed search-factors. Since the strength of the declarative representation of these two factors decayed when the OTS instructions were not available, retrievals of these factors were less likely to be included in the visual search (see Fig. 2).

The pattern of results suggests that it is equally important to predict when to provide the appropriate OTS instructions to facilitate short-term and long-term performance. In our study, the criterion under which an OTS instruction was given was fixed (i.e., when participants hooked a track that was 20% less important that all the unidentified tracks), but our results suggest that it is important to have a flexible model-based training system that adapts to different stages of learning of the trainee. For example, it is possible to monitor the proceduralization process to determine when additional OTS instructions should be given more often (when the instructions have not been fully proceduralized) or less often (when the instructions have already been fully proceduralized, and OTS instructions may slow down short-term performance without facilitating long-term performance).

9. General discussion

We found that participants were able to learn from over-the-shoulder instructions and paid attention to more important tracks. To understand how people learned from

[7] The use of range as a factor even in the no-instruction group provided support for the assumption of the model that people tend to select the first track near the ANZIO.

[8] In ACT-R utility value may decay through a similar decay process in chunk activation, but we decided not to use them as they add additional complexity to the model and it is not the focus of our current modeling effort.

these instructions, we developed a mechanism that allows the system to deliver OTS instruction and the model to learn from the instructions. Instructions were represented as declarative memory traces. Repeated instructions strengthened these memory traces to make them more accessible when needed. A mechanism is constructed that allows visual search to be incrementally improved and declined based on the strengths of the memory traces of the top-down instructions. The ability to search for important tracks improved as the strengths of these memory traces increased, and the production compilation process created and strengthened new production for better search. However, when these instructions were not fully proceduralized, the strengths of the memory traces of the instructions would decay, leading to the decline in performance.

The treatment of the OTS instructions is the same as the initial instructions, except that (a) they take time away from the processing, (b) their effects tend to decay quite rapidly in the initial phase of training, and (c) their effects on training diminishes as the OTS instructions are proceduralized in the late stages of training. We realistically model the slow learning of the OTS instructions whereas we assume that the initial instructions are well encoded. Intuitively, one might speculate that OTS instructions might be easier to learn as they are presented right in the exact same situation. As least from our data, although the OTS instructions apparently had a large impact on short-term performance, they did take time away from the task and their effect seemed to decay quite rapidly before they were fully proceduralized.

Model-based training systems have been successfully applied in relatively static task domains such as algebra or geometry. The current line of work attempts to transfer the technology of model-based training systems out from the protected classroom or laboratory to a real-world, complex, and dynamic task. The task requires all forms of learning and all buffers in ACT-R to interact and thus is a good test bed for the architecture. Although ACT-R seems to be a good learning system capable of tracking the skill acquisition process, it does not have any "built-in" mechanisms for interpreting or comprehending instructional texts that other models such as those based on the Construction-Integration theory (Kintsch, 1988) are capable of. Indeed, the comprehension often seems to be an integral part of skill acquisition or planning (e.g., Doane et al., 2000; McNamara et al., 1996) that our system has not yet focused on. Future integration of these capabilities by combining the merits of these systems with the current system will be promising.

The current system is of course far from a perfect training system. To begin with, we have provided only a model of aggregate behavior, which is known to be good for capturing general cognitive phenomena, as we intended to do here. The limitation of model that fits aggregate data is that the conclusion only applies to the general trend of acquisition. Thus, we are not able to infer, for example, why an individual may learn better or worse than others. To fully utilize the capabilities of the system, it is desirable to develop models that closely track the skill acquisition process of individuals. However, fitting individual data requires a much larger sample size and a much more careful control of variables that contribute to individual differences (e.g., Engle et al., 1999). For example, if factors such as working memory capacity are believed to be the underlying causes of individual differences in learning, the general approach is to first conduct independent measures of the factors for each individual. The major challenge is how one can construct models and search for parameters that reflect the effects of different levels of the factors to learning performance. The advantage of this approach is that in addition to predicting individual learning performance, it also provides explanations of why certain individuals may, for example, be better learners than others based on the factors identified during the analyses of individual differences. The intelligent training system can therefore provide instructions that are specific to the "cognitive profile" of the individual. This feature, although desirable, clearly requires more careful empirical testing and modeling effort in the future.

The concept of human-computer symbiosis started around the 1960s and has since set an agenda to give "machine organic qualities at the same time that a parallel program in psychology mechanized our understanding of our minds and bodies" (Crowther-Heyck, 2005). As the theme of this special issue, symbiotic performance between humans and intelligent system demands better communications that enable humans and computers to interact in real time via various techniques. In this paper, we show that a combination of a cognitive model of skill acquisition and a simple action protocol tracer provides close connection between the learner and the training system by tracking and interpreting actions in real-time. Through the process of tracking and interpretation, the intelligent system communicates with the learner by inferring functional goal and intention of the learner, and provides real-time instructional materials to augment the skill acquisition process. We hope that the work described in this article has suggested some of the methods and insights that may encourage future research on better communications between humans and machines.

Appendix A

Rules for the GT-ASP Experiment

(In parenthesis are the exact instructions that were given to the ACT-R model, the underlined are the crucial information that was left out for subjects to fill in during the test.)

1. The task is to identify unidentified tracks. Unidentified tracks are half squares with vectors emanating from them. One should hook (click on) such tracks and then go through the sequence of identifying them. (To identify-tracks first look-for a track that is "half-square" then hook the track then id-sequence the track and then repeat.)
2. One way to identify a track is to confirm that it is flying at a commercial altitude and speed and then record it as friendly primary id and non- military air id. (To id-sequence a track first altitude-test then speed-test and then record it as "friend" "non- military".)
3. The other way to identify a track is to request its EWS identity, and then classify the track according to that identity. (To id-sequence a track first ews the track for a ews-signal and then classify the track according to the ews-signal.)
4. To confirm that a plane is flying at the commercial altitude, look in the upper left, search down for "alt", read the value to the right, and confirm that it is more than

25,000 and less than 35,000. (To altitude-test first seek "upper-left" and then search-down for "alt" at a location then read-next from the location a value then check-less 25000 than the value and then check-less the value than 35000.)

5. To confirm that a plane is flying at the commercial speed,look in the upper left, search down for "speed", read the value to the right, and confirm that it is more than 350 and less than 550. (To speed- test first seek "upper-left" and then search-down for "speed" at a location then read-next from the location a value then check-less 350 than the value and then check-less the value than 550.)
6. To request the EWS identity of a track, select the "ews" key, then select "query sensor status" key, and encode the value that you are told. (To ews a track for a ews-signal first select "ews" then select "query sensor status" and then encode-ews the ews-signal.)
7. To classify a track whose EWS identity is ARINC record it as "friendly" primary id and "non- military" air id. (To classify a track according to a ews-signal first match the ews-signal to "arinc564" and then record it as "friend" "non- military".)
8. To classify a track whose EWS identity is APQ record it as hostile primary id and strike air id. (To classify a track according to a ews-signal first match the ews-signal to "apq" and then record it as "hostile" "strike".)
9. To classify a track whose EWS identity is APG record it as friendly primary id and strike air id. (To classify a track according to a ews-signal first match the ews-signal to "apg" and then record it as "friend" "strike".)
10. To classify a track whose EWS identity is negative treat it as unclassifiable. (To classify a track according to a ews-signal first match the ews-signal to "negative" and then mark-node the track.)
11. To record a primary id and a secondary id select the following sequence of keys: "track manager", "update hooked track", "class/amp", "primary-id", the primary id, "air-id", the air-id, "save" and then you have succeeded. (To record a primary-id and a air-id first select "track manager" then select "update hooked track" then select "class/ amp" then select "primary id" then select the primary-id then select "air id amp" then select the air-id then select "save changes" and then success.)
12. To select a key, find where it is in the menu and hit the corresponding F-key. (To select a option first find-menu the option at a location and then hit-key corresponding to the location.)
13. To find where an item is in the menu, look to the lower left and search to the right for the term. (To find-menu a option at a location first seek "lower-left" and then search-right for the option at a location.)

References

Anderson, J.R., 1983. The architecture of cognition. Harvard University Press, Cambridge, MA.
Anderson, J.R., Bothell, D., Byrne, M.D., Douglas, S., Lebiere, C., Qin, Y., 2004. An integrated theory of the mind. Psychological Review 111 (4), 1036–1060.

Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R., 1995. Cognitive tutors: lessons learned. The Journal of the Learning Sciences 4, 167–207.

Doane, S.M., Sohn, Y.W., McNamara, D.S., Adams, D., 2000. Comprehension-based skill acquisition. Cognitive Science 24 (1), 1–52.

Crowther-Heyck, H., 2005. Mind and network. IEEE Annals of the History of Computing 27, 103–104.

Douglass, S. 2004. Using the plastic framework to augment scenario-based training systems with instructional agents. In: Proceedings of the 48th Annual Meeting of the Human Factors and Ergonomics Society, pp. 2524–2528.

Engle, R.W., Tuholski, S.W., Laughlin, J., Conway, A.R.A., 1999. Working memory, short-term memory and general fluid intelligence: a latent variable model approach. Journal of Experimental Psychology: General 128, 309–331.

Fu, W.-T., 2001. ACT-PRO action protocol analyzer: a tool for analyzing discrete action protocols. Behavior Research Methods, Instruments, and Computers 33 (2), 149–158.

Fu, W.-T., Anderson, J.R., 2006. From recurrent choice to skill learning: a reinforcement-learning model. Journal of Experimental Psychology: General 135, 184–206.

Fu, W.-T., Gray, W.D., 2000. Memory versus perceptual-motor tradeoffs in a blocks world task. In: Proceedings of the Twenty-second Annual Conference of the Cognitive Science Society. Erlbaum, Hillsdale, NJ, pp. 154–159.

Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H., Ventura, M., Olney, A., Louwerse, M.M., 2004. Auto tutor: a tutor with dialogue in natural language. Behavioral Research Methods, Instruments, and Computers 36, 180–193.

Gray, W.D., Fu, W.-T., 2004. Soft constraints in interactive behavior: the case of ignoring perfect knowledge in-the-world for imperfect knowledge in-the-head. Cognitive Science 28 (3), 359–382.

Gray, W.D., Schoelles, M.J., (2003). The nature and timing of interruptions in a complex cognitive task: empirical data and computational cognitive models. In: Proceedings of the 25th Annual Meeting of the Cognitive Science Society. pg 37.

Hill Jr., R.W., Johnson, W.L., (1993). Designing an intelligent tutoring system based on a reactive model of skill acquisition. Proceedings of the International Conference on AI and Education, Edinburgh, 1993.

Hodge, K.A., Rothrock, L., Kirlik, A.C., Walker, N., Fisk, A.D., Phipps, D.A., Gay, P.E. (1995). Trainings for tactical decision making under stress: towards automatization of component skills. (HAPL-9501). Atlanta, GA: Georgia Institute of Technology, School of Psychology, Human Attention and Performance Laboratory.

Kintsch, W., 1988. The use of knowledge in discourse processing: a construction-integration model. Psychological Review 95, 163–182.

Laird, J.E., Newell, A., Rosenbloom, P.S., 1987. Soar: an architecture for general intelligence. Artificial Intelligence 33 (3), 1987.

McNamara, D.S., Kintsch, E., Songer, N.B., Kintsch, W., 1996. Are good texts always better? interactions of text coherence, background knowledge, and levels of understanding in learning from text. Cognition and Instruction 14, 1–43.

Meyers, D.E., Kieras, D.E., 1997. A computational theory of executive cognitive processes and multiple-task performance. Part 1. Basic mechanisms. Psychological Review 104, 2–65.

Newell, A., 1973. Production systems: models of control structures. In: Chase, W.G. (Ed.), Visual information processing. Academic Press, New York, pp. 463–526.

Newell, A., 1990. Unified theories of cognition. Harvard University Press, Cambridge, MA.

Ritter, S., Koedinger, K.R., 1997. An architecture for plug-in tutoring agents. In: Journal of Artificial Intelligence in Education, 7 (3/4), 315–347. Charlottesville, VA: Association for the Advancement of Computing in Education.

Rescorla, R.A., Wagner, A.R., 1972. A theory of Pavlovian conditioning: variations on the effectiveness of reinforcement and nonreinforcement. In: Black, A.H., Prokasy, W.F. (Eds.), Classical conditioning:II. current research and theory. Appleton-Century-Crofts, New York, pp. 64–99.

Sleeman, D., Brown, J.S., 1982. Intelligent Tutoring Systems. Academic Press, New York.

Sutton, R.S., Barto, A.G., 1981. Toward a modern theory of adaptive networks: expectation and prediction. Psychological Review 88, 135–170.