SimPilot: An exploration of modeling a highly interactive task with delayed feedback in a multitasking environment

Michael Schoelles (schoem@rpi.edu)
Cognitive Science Department, Rensselaer Polytechnic Institute
Troy, NY 12180 USA

Wayne D. Gray (grayw@rpi.edu)
Cognitive Science Department, Rensselaer Polytechnic Institute
Troy, NY 12180 USA

Abstract

Taxiing an airplane at a major airport requires the pilot to interact the world outside the cockpit, the instrumentations within the cockpit, and the co-pilot. As many actions require time to pass before their outcome can be evaluated, the pilot must have an approximate sense of how much delay should occur before the outcome of an action can be evaluated. Finally, taxiing is a paradigmatic example of multitasking. These three ingredients (a) a high level of interaction with dynamic task environments, (b) a sense of time, and (c) multitasking, present challenges for theories of cognition and the building of process models of taxiing. We describe a model, *SimPilot*, its initial validation, and its implications for cognitive theory.

Keywords: Multitasking; interactive behavior, delayed feedback, threaded cognition; cognitive control; task switching.

Introduction

A good applied problem drives basic science and a good theory can be useful, to paraphrase Newell (Newell & Card, 1985) and Lewin (1951). We believe such a relationship exists between taxiing an aircraft and cognitive theory. A task analysis of taxiing reveals that multiple cognitive, perceptual, and motor actions are preformed in parallel. The pilot must listen for commands and respond to commands from the ground controllers, while navigating the complex layout of airports. The pilot must steer and turn the plane while monitoring the aircraft speed. These and the other tasks require pilots to manipulate instruments in the cockpit while attending to signs and other planes outside the cockpit. In addition, key steps in taxiing require verbal and gestural contact with the co-pilot. The complexity of the Boeing 737-800 cockpit is evident in Figure 1.

In summary, taxiing is a paradigmatic example of multitasking where each task has its own subtasks that must be interleaved with those of the other tasks. It is exceedingly interactive. Not only does the pilot create change in the external and internal environment but many changes to which the pilot must respond arise from external factors. Finally, unlike most tasks modeled by cognitive science[1], changes produced by pilot actions may not be immediately apparent but requires the pilot to maintain some sense of passing time and some expectation for when the results of his or her actions should become apparent.

SimPilot models the cognition and behavior of an airline pilot taxiing a Boeing 737-800. The mechanisms for multitasking that we use are Salvucci and Taatgen's theory of *threaded cognition* (2008; 2010), which at the moment is a candidate architectural mechanism within the ACT-R cognitive architecture (Anderson, 2007; Anderson, et al., 2004). We view SimPilot as an exploration of the strengths and weaknesses of the theory of threaded cognition as well as a potential tool for aviation psychology.

An open question for cognitive architectures is whether multitasking should be considered an architecture feature or a strategic adaptation that is driven by the accommodation of more basic architectural features to the demands of the task environment. Hence, besides being of applied interest, SimPilot may shed light on a key theoretical question.

Figure 1. The Boeing 737-800 cockpit.

Indeed, we suggest that SimPilot is the most comprehensive model yet developed using threaded cognition. Unlike prior models that are dual-threaded, SimPilot models three threads and uses the entire ACT-R 6.0 architecture to perform the taxiing application. In this paper, the next section will provide a brief overview of ACT-R 6.0 with threaded cognition. Then previous work within ACT-R to model multitasking performance is discussed. The SimPilot model is then described. We

[1] But not all tasks, see Anzai (1984), for an exception.

discuss the predictions the model made about pilot performance and how those predictions fared in an empirical evaluation. We conclude with an evaluation of threaded cognition as a mechanism to perform multitasking within a cognitive architecture. We end with a few words about the use of the simulated task environment, X-Plane™, for cognitive science research and computational cognitive modeling.

ACT-R 6.0

ACT-R 6.0 (Anderson, 2007) is an embodied cognitive architecture that has perceptual and motor components along with cognitive processing, memory, and control components. The perceptual and motor components enable SimPilot to operate user interfaces by passing the interface software the same commands passed by the input devices used by humans. As the SimPilot is a cognitive, not an artificial intelligence model, its input commands mimic the speed and accuracy of human users.

The ACT-R architecture requires the modeler to specify two types of knowledge. Declarative knowledge specified by the modeler represents background knowledge required to perform a certain task. The declarative knowledge is represented as chunks. A chunk has a type, which serves to specify the structure of the chunk. The chunk structure is composed of named slots, which hold values. Declarative chunks are stored in the Declarative Memory Module. Time to retrieve an item from memory varies as a function of the recency and frequency of that item's occurrence (Schooler & Anderson, 1997; Sims & Gray, 2004). Like humans, errors can occur in the memory retrieval process due to random fluctuations (noise) in memory strength or activation (Sims & Gray, 2004). Either the wrong chunk is retrieved or the intended chunk is not "strong" enough to be remembered.

The second type of knowledge specified by the modeler is procedural knowledge in the form of pattern matching productions. Productions specify how a certain task is done. A production consists of a set of constraints that must be satisfied before the actions specified by the production can be executed. Productions are stored in the Procedural Module. ACT-R checks every 50ms (human time) all of its productions and executes one of the productions whose pattern is matched. If more than one production can execute then ACT-R chooses the one it calculates would be the most useful at this time. This serial execution is not as constraining as it might seem and has been shown to be as accurate at simulating fine-grained human behavior as architectures that allow parallel firing of productions (Byrne & Anderson, 2001). If a production could fire but did not because another one had a higher utility, chances are that in 50 ms it will be able to fire. The productions are intended to represent the fine-grained procedural steps that are executed to perform some task. ACT-R adds noise to the utility calculation to simulate the variability in time and performance that humans make.

ACT-R maintains simulated human time in that time for ACT-R processes and actions are set to the theoretical times for the corresponding human events such as shift of visual attention, or memory retrieval. When the model does a task ACT-R produces a trace that includes the action taken and a time stamp. The trace allows model performance to be compared with human performance.

The perceptual components of ACT-R allow the model to see and hear. In common with the human brain, the visual component has *where* and *what* paths (Findlay & Gilchrist, 2003). The *where* path allows the model to detect features of an object such as color, size, and shape at a 2-D location in space. The *what* path moves visual attention to that location to encode the object with those features. ACT-R *hears* in much that same way that it sees in that sound events are detected and auditory attention is invoked to encode those sounds. By encoding objects and sounds in the environment the visual and auditory components add new declarative knowledge to the model. The motor component is the model's *hands* and *voice*. The manual component is capable of moving and clicking the mouse. Movement times are a based on Fitts' Law (Fitts, 1954). The vocal module is capable of speaking text and subvocalization (see, e.g., Huss & Byrne, 2003).

The *imaginal* component of ACT-R is intended to hold intermediate representations required in solving a problem or performing some task. New declarative chunks can be added by this component. The *temporal* component maintains an internal clock. The *goal* component in hold chunks that guide task execution. For the model presented in this paper the default goal component is replaced with a module that implements a form of threaded cognition (Salvucci & Taatgen, 2008) that implements the multitasking required for the taxiing task.

Threaded Cognition

The current architectural candidate for multitasking in ACT-R is threaded cognition (Salvucci & Taatgen, 2008, 2011); an integrated theory of concurrent multitasking. Multitasking is defined as doing 2 or more tasks at once. A thread is sequence of processing steps coordinated by a serial procedural resource and executed across perceptual and motor resources. The key claims of threaded cognition are that multiple active goals can exist. Associated with each goal is a block of procedural processing. Processing conflicts can exist for procedural, declarative, perceptual, and motor resources. A thread will grab a resource if it needs the resource and the resource is available. It will release the resource when no longer needed. According to Salvucci and Taatgen, cognition favors the least recently processed thread. Declarative retrievals can be converted to hard coded productions over time thus reducing both declarative and procedural resource conflicts.

Background

Salvucci (Salvucci et al., 2006) explored interleaving task segments during a *discrete driving* task. Subjects used a

keyboard to steer a vehicle while entering navigation information as a second task. In particular they investigated how changing the timing characteristics of the driving task affected the interleaving of the tasks. To do this they used the temporal module of ACT-R and threaded cognition. They developed a top down and a bottom up model of cognitive control. In the bottom-up model, events (that is, changes to ACT-R's perceptual and temporal buffers) determine when task switching will occur. In the top-down model, the general executive (an early version of threaded cognition) interleaves the two tasks. Both models change the time interval for switching based on whether recent switches were early, late or on time.

Borst and Taatgen (2007) extended the threaded cognition concept that peripheral resources and declarative memory are shared between processes without executive intervention to problem representations, which are maintained in the imaginal buffer. They modified the discrete driving task of Salvucci (2006) to devise two tasks, one hard and one easy, in which the participants had to keep track of the problem state. The experiment and model they developed did show extra interference in task performance when both tasks needed the imaginal buffer.

Veksler (2011) used the latest implementation of threaded cognition in ACT-R 6.0 in a decision making task to monitor a task event while searching a display. This task is a spin-off of the Argus task (Schoelles, 2001) in which 20 targets appear on a radar screen. Each target has an assigned threat value, which the participant can acquire by clicking on the target. The task objective is to find the target with the largest threat value from a table of six alternatives, which is displayed on the right side of the screen. The degree of difficulty in acquiring the treat value is implemented via a lockout period, which is the time from clicking on the target to the actual display of the threat value. The lockout period was a between-subject condition that varied from 0 seconds to 8 seconds. Without any perceptual constraints her initial model switched many more times than human participants. When the perceptual constraint that the monitor task will only be initiated if model has not found a pre-attentive feature during the search thread was implemented, then the model did a much better at matching human switch rates.

Zemla (Zemla et al., 2011) has developed an ACT-R model of the taxiing task using the X-Plane simulation. The model does not use threaded cognition, since the focus of the model to produce a high-fidelity model of the turning and steering the plane while taxiing. The model is quite successful in modeling these subtasks. SimPilot does not steer or turn with the accuracy of this model but is more concerned with the interactive, multitasking aspects of the taxiing task.

SimPilot Description

The SimPilot model was developed as a proof-of concept system that intended to show that cognitive modeling can be applied to the evaluation of new technologies in aviation that are intended to increase runway safety. The system consisted of the SimPilot model and ACT-R 6.0 running on one system, and the X-Plane simulation of a Boeing 737-800 running on the same or different computer. The communication between SimPilot and X-Plane simulator was via TCP/IP. Other aircraft running X-Plane on other computers simulated ground traffic at the airport. The X-Plane software provides several different interface options. One option is through a Software Development Kit (SDK). The user develops a plugin following the specifications of the SDK to access data variables used by X-Plane. Most of the data values that are displayed in the cockpit can be accessed both for reading and writing in this manner. The X-Plane system polls the plugin for requests to read and write these data values. For example, the ground speed of the aircraft can be read or the frequency of the radio can be read or set in this manner.

The scenario modeled by SimPilot begins with clearance from Air Traffic Control at the Dallas-Ft.Worth Airport, to taxi from the terminal, via a prescribed route to the *hold short* area of the runway. Once there, SimPilot must wait for clearance, then move onto the runway, and takeoff. Instructions and flight information can be given by Air Traffic control at anytime along the route. The route could involve several taxiways to reach the runway and other simulated aircraft. Concurrent subtasks include taxiing, monitoring the speed of the aircraft, maintaining situation awareness, and steering.

SimPilot Structure

SimPilot specifies visual-location chunks and visual-object chunks for the cockpit instruments. This file is read when the model is loaded. When the model moves visual attention to the location of an instrument the corresponding visual-object chunk is created.

ACT-R models are goal-driven, that is, task control is specified in declarative chunks that will serve as the current task goal. SimPilot specifies task goals for a number of subtasks, such as power-up, steering, turning, tune radio, switch lights, monitor speed etc. Since SimPilot uses multitasking, a task control chunk that specifies the subtasks that can execute together is also specified.

Productions have three structures. One structure does not have a goal but reacts to new information from the external environment. Examples include commands from ground control or comments from the co-Pilot. The structure of the task control productions specify what tasks can run concurrently. Third, regular productions have a goal with either a state slot or a time constraint slot.

SimPilot Task Control Flow

The first version of SimPilot did not use threaded cognition, but attempted to implement multitasking by using ACT-R's default ACT-R, one-level deep, goal buffer in several different ways. It was found that this switching was either not very cognitively plausible or took too much time. In one approach, the old goal was stored in a slot in the new goal and retrieved directly from that slot when it was necessary

to switch back. This approach is not cognitively plausible or very flexible with more than two goals. Another approach is to retrieve old goals through declarative memory (Altmann & Gray, 2008) but this approach did not meet the time requirements of a highly interactive environment requiring immediate actions. Threaded cognition offered a good alternative although all models using threaded cognition up to this point have been far simpler than the complexity required for taxiing. Task analysis also showed that the number of tasks that are being done at one time varies. The threaded cognition goal module maintains a set of goals. The goal buffer is used as part of threaded cognition and in accordance with Taatgen's Minimal Control Principle (Taatgen, 2007), using a goal state slot is kept to a minimum. The control constraints come from the availability of the perceptual modules. In addition, the temporal module and buffer are used to monitor events at specific intervals, which also provides a form of control.

We define a subtask (of taxiing) to be a set of one or more goals to be executed using threaded cognition. That is, the goal module is responsible for switching the goal buffer between these goal chunks. The subtask control chunks are linked together though a slot which contains the next subtask to be executed. When the current goal set has accomplished its part of the task a subtask control chunk is made the current goal, which retrieves the chunk specifying the next multitasking set.

In SimPilot, a thread maintains control by either not clearing the perceptual buffers or having the imaginal buffer maintain a representation unique to the thread. For the most part a thread can only start executing if the perceptual and imaginal buffers are clear. A thread gives up control by clearing these. As the model was being developed it was realized that communication between threads is sometimes required. Normally in ACT-R communication between productions is done through the imaginal buffer, but in SimPilot the imaginal buffer is thread specific so this pointed to the need for another buffer, which can be considered as a extended imaginal buffer that is common to all threads. In SimPilot this is called the *situation buffer*.

SimPilot Steering When humans steered X-Plane, they used a joystick that had a nose wheel control capability. For the model to steer X-Plane several options are available. Early on in the project the joystick was configured to change the yoke pitch, roll, and yaw, so initially the option that sets the SDK variables for the yoke pitch, roll, and yaw was implemented. When the joystick was reconfigured to manipulate the nose wheel, the option that the mouse acts as the joystick and changes the nose wheel in the same way as the human subjects was implemented. Also, steering requires both perception and manual operations. SimPilot looks at the heading display on the cockpit to monitor the current heading and looks at a point on the windshield, which is the target for the movement. It uses the temporal module to keep from responding too fast to course corrections.

SimPilot Turning Turning is similar to steering, but in turning the plane goes from a beginning heading to a final heading, and the current heading changes rapidly. Also, turns decreases the momentum of the aircraft so human pilots often increase thrust during the second half of the turn. In the first half of the turn SimPilot steadily increases the deflection of the nose wheel in the direction of the turn, it then brings it back to zero deflection in the second half, always trying to keep the rate of turn below 5 degrees/second. SimPilot looks at a point on the windshield and moves the cursor to that location.

SimPilot Parameters The model is predictive in the sense that it was not fit to data. The standard ACT-R parameters were used. It would be very hard to fit the model to data for individual parameters due to the complexity of the task. Likewise, as the model requires the X-Plane simulation and can only be run in real time, each model run takes 5-10 minutes. This real-time constraint makes it nearly impossible to do the 100's or 1000's of model runs that most model fitting requires.

SimPilot Representation of Environment

Interactive models like SimPilot are heavily dependent on a good representation of the environment. The cockpit that X-Plane provides (as shown in Figure 1) is very complex and contains many user interfaces objects that are not the usual Human Computer Interaction (HCI) type of objects. For example, tuning the radio, which is one of the capabilities of the model, requires an interactive routine of 22 productions. The radio has an inner and outer dial. The outer dial determines the integer portion of the frequency and the inner dial controls the decimal portion. The radio has an active indicator and a standby indicator. The model sets the stand-by indicator and then presses a button to switch the frequency entered to the active display.

To navigate the airport or follow a route, the locations of airport signs is required. Again X-Plane does not provide an automatic way to encode this information. We used Google Earth to obtain some of the taxiway and runway locations. KML files were exported from Google Earth and the MSS contains some code to parse these files.

SimPilot Performance Measures

Table 1 shows the number of visual attention shifts and the number of subtask switching for the speed monitoring, navigation monitoring and steering threads for two taxiing tasks. These results were extracted from the ACT-R trace files produced for each run. Due to lack of eye data we cannot compare these to human saccades and fixations.

Table 1. Model Attention and Subtask Shifts.

Measure	Taxi to 35L (564 secs)	Taxi to 35R (738 secs)
Visual Attention Shift	1042 (1.8/s)	1362 (1.8/s)
Speed Monitoring	381 (0.7/s)	409 (0.6/s)
Navigation Monitoring	432 (0.8/s)	568 (0.8/s)
Steering	432 (0.8/s)	566 (0.8/s)

SimPilot Validation

Human data was collected from 6 pilots with varying degrees of experience. The model has only gone through an alpha where no model parameters were fit to the data, so the model data presented in Table 2 (Jungemann, 2011) can be interpreted as model predictions. In fact the model was only run in the complete system configuration once, which did expose several problems.

Table 2. Performance measurements and results

Performance Measure	Human Mean/SD	Model Mean/SD
Average Taxi Speed	9.8/1.4	6.5/0.7
Maximum Taxi Speed	15.3/2.6	8.7/0.5
Average Throttle Setting	0.13/0.03	0.11/0.002
Maximum Throttle Setting	0.46/0.21	0.15/0.01

The data shows that this version of the model is far from being an expert pilot, but several factors make quantitative comparison of model and human data difficult. The braking capability of the human pilots was very different than the model. The humans had pedals that were used to brake the plane, but ACT-R has no foot motor module or interface to the pedals so the model performed braking through adjusting a X-Plane variable. The human pilots used a joystick to steer the aircraft via the nose wheel of the plane. The model could not control the joystick directly in X-Plane but used the cursor to control the nose wheel.

Some of the measures that we could compare model versus human performance are shown in Table 2. The average taxi speed is the aircraft ground speed in nautical miles per hour, calculated from start of taxi to the hold short area of the runway. The maximum taxi speed is the highest ground speed attained on taxiway. The average throttle setting is X-Plane data value that ranges from Idle to Full. The maximum throttle setting is the highest throttle setting from start to hold short.

Discussion

Since we do not have actual data on the task switching behavior of human pilots, novice or expert, taxing with X-Plane, our evaluation of threaded cognition is from an engineering perspective. By itself the implementation of threaded cognition is underspecified for multitasking in this environment. The main issue for cognitive engineering is how much does threaded cognition shift the burden of task switching from the modeler to the architecture. In the ACT-R architecture without threaded cognition the modeler must make an explicit request of the goal module to switch goals. With threaded cognition the goal module, if two or more goals match, will allow only one production with a goal of that type to execute at a time, since it will be placed at the end of the queue after its production executes. This will be fine in situations where this type of alternate behavior is required, but in most situations other factors determine task behavior. For example, pilots do not alternate between checking the speed of the plane and looking at the center line. The pilots check their position on the taxiway much more than checking the speed. The speed is checked at periodic intervals, which is implemented in SimPilot by the temporal module.

In many cases, the first production of a thread is a check for some condition, for example, is the speed of the aircraft within certain limits. If not, then the function of the thread is to correct that condition as fast as possible. In these cases the threat should not be interrupted. With the current implementation, it is up to the modeler to code this into the productions. So while some of the task control has been shifted to the architecture, much of it must still be done by the modeler.

In developing the model, it became evident that decomposing the taxiing task into subtasks required that each subtask maintain its own representation in the imaginal buffer to hold the knowledge unique to that subtask. For example, monitoring speed requires knowledge about speed limits while navigating requires knowledge about position. Global data, such as where you are on the route to the runway, is data shared by different subtasks and need to be held in a shared buffer. In SimPilot a global buffer called the situation buffer was created. This buffer can be thought of as an imaginal buffer for the entire task. It has the same modeling considerations as regular imaginal buffers such as how long should the data persist before some it should be practiced, etc.

The interface to the X-Plane environment initially seemed promising but has not worked as hoped. For the cockpit interface, SimPilot needs to know the x and y pixel location, name, color and size of all the instruments. X-Plane does provide the names and the x and y pixel locations for some of the instruments but not for the majority of them. This information has to be hand calculated which is a very labor-intensive process. The reactiveness of the model to changes in the plane's data values is constrained by the polling interface. X-Plane does provide a datagram interface, which would allow faster interactions between the model and X-

plane, but this interface is not well-documented nor guaranteed not to change.

In order to be able to make comparisons between SimPilot and human pilots, an analysis of where humans are looking while they are doing this task is essential. The next step in the development of SimPilot is to collect eye data on human multitasking behavior in this task and use the results improve the model.

To simulate the multitasking required to taxi a Boing 737-800 at a major airport, the SimPilot model uses all the components of ACT-R with an implementation of the theory of threaded cognition. This effort has both theoretical implications for ACT-R and threaded cognition and is a start on solving the important applied problem of the effects of multitasking on pilot performance.

Acknowledgments

This work was supported by NASA Ames Research Center through a Small Business Innovative Research Contract (Contract No. NNX09CB45C) awarded to Aptima, Inc. All opinions expressed in this article are those of the authors and do not necessarily reflect the official opinion or position of Rensselaer Polytechnic Institute, Aptima, Inc., or NASA. We would like to thank Dr. James Johnston and Dr. Michael Feary at NASA Ames Research Center for funding this research. We also would like to thank Gabe Ganberg, Jeff Jungemann, Joe Thoreson and Jamie Estock of Aptima, Inc.

References

Altmann, E. M., & Gray W. D. (2008). An integrated model of cognitive control in task switching.. Psychological review. 115(3), 602-39.

Anderson, J. R. (2007) How Can the Human Mind Occur in the Physical Universe? New York: Oxford University Press.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. L. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036-1060.

Anzai, Y. (1984). Cognitive control of real-time event-driven systems. *Cognitive Science, 8*, 221–254.

Byrne, M. D., & Anderson, J. R. (2001). Serial modules in parallel: The psychological refractory period and perfect time-sharing. *Psychological Review, 108*(4), 847–869.

Byrne, M. D., & Kirlik, A. (2005). Using computational cognitive modeling to diagnose possible sources of aviation error. *International Journal of Aviation Psychology, 15*, 135-155.

Byrne, M. D., Kirlik, A., & Fleetwood, M. D. (2008). An ACT-R approach to closing the loop on computational cognitive modeling: Describing the dynamics of interactive decision making and attention allocation. In D. C. Foyle & B. L. Hooey (Eds.), *Human performance modeling in aviation* (pp. 77-104). Boca Raton, FL: CRC Press.

Findlay, J. M., & Gilchrist, I. D. (2003). *Active vision: The psychology of looking and series*. New York: Oxford University Press.

Fitts, P. M. (1954). The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology, 47*(6), 381–391.

Huss, D. G., & Byrne, M. D. (2003). An ACT-R/PM model of the articulatory loop. In F. Detje, D. Dörner & H. Schaub (Eds.), *Fifth International Conference on Cognitive Modeling* (pp. 135-140). Bamberg, GE: Universitas-Verlag Bamberg.

Jungemann, J. (2011).Aptima Final Report: Computational Model and Measurement Tool for Evaluating the Design of Flight Deck Technologies II.

Lewin, K. (1951). Field theory in social science. New York: Harper Row.

Newell, A. & Card, S. K. (1985). The prospects for psychological science in human-computer interaction. Human-Computer Interaction 1, no 3, 209-242.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review, 115*(1), 101-130.

Salvucci, D. D., & Taatgen, N. A. (2011). *The multitasking mind*. New York: Oxford University Press.

Schoelles, M. J., & Gray, W. D. (2001a). Argus: A suite of tools for research in complex cognition. Behavior Research Methods, Instruments, & Computers, 33(2), 130-140.

Schooler, L. J., & Anderson, J. R. (1997). The role of process in the rational analysis of memory. *Cognitive Psychology, 32*(3), 219-250.

Sims, C. R., & Gray, W. D. (2004). Episodic versus semantic memory: An exploration of models of memory decay in the serial attention paradigm. In M. C. Lovett, C. D. Schunn, C. Lebiere & P. Munro (Eds.), *6th International Conference on Cognitive Modeling* (pp. 279-284). Pittsburgh, PA: Carnegie Mellon University/University of Pittsburgh.

Taatgen, N. A. (2007). The minimal control principle. In W. D. Gray (Ed.), *Integrated models of cognitive systems*. New York: Oxford University Press.

Veksler, B. (2011) Modeling the Visual Search Process. Unpublished Doctoral Thesis, Rensselaer Polytechnic Institute

Zemla, J.C., Ustun, V., Byrne, M.D., Kirlik, A., Riddle, K., & Alexander, A.L. (2011).An ACT-R Model of Commercial Jetliner Taxiing.Proceedings of the 55th Human Factors and Ergonomics Society.