

Modeling Patterns of Performance in Air Traffic Control Operations

Roger W. Remington¹, Seung Man Lee², Ujwala Ravinder²,
Ben Willems³, Michael Freed⁴

Summary

Computational modeling of human performance can help anticipate the response of air traffic controllers to changing roles, but its use has been hindered by the difficulty in constructing models in complex domains. We have previously reported a method for constructing extended behavioral sequences automatically from a simple underlying cognitive architecture [1,2]. Here we report on efforts to extend that method to the complex domain of air traffic control operations.

Introduction

Automation is changing the role of air traffic controllers, generating the need to evaluate human performance under new concepts of operation. Computational modeling of the human air traffic controller could be valuable in anticipating the response of the controller to new automation, displays, or procedures. Unlike current empirical evaluation methods, modeling and simulation permits the exploration of a large number of scenarios and designs. To generate accurate predictions (zero-parameters fit) from a model that supports reuse across applications, it is necessary to construct detailed models of the fundamental mental operations underlying task performance (e.g., [4,5]). Models at this level of granularity are difficult and time consuming to develop, limiting their applicability to complex domains. We have previously reported success in automating the construction of detailed mental models using the Apex computational architecture [6] to automate demanding aspects of constructing models in the CPM-GOMS cognitive architecture [1,2]. In this approach, behavioral sequences are constructed from *templates*, which model the elementary cognitive, perceptual, and motor operations underlying primitive task behaviors, such as typing, moving and clicking a mouse, or speaking a string of words. Because such behaviors occur in many domains, templates can be stored in libraries and reused from one model to the next. Here we report on an extension of the

¹NASA Ames Research Center, MS 262-4, Moffett Field, CA 94035, (650) 604-6243, rremington@mail.arc.nasa.gov

²NASA Ames Research Center & San Jose State University Foundation, {smlee, uravinder}@mail.arc.nasa.gov

³William J. Hughes Technical Center Atlantic City International Airport, NJ 08405, ben.willems@faa.gov

⁴NASA Ames Research Center & University of West Florida, mfreed@mail.arc.nasa.gov

template approach that begins to address multitasking, a common feature of complex domains. We first discuss the requirements for modeling multitasking, then compare predictions of an Apex model of air traffic controller performance when executing sector handoffs to data from a simulation conducted by the Federal Aviation Administration.

Modeling Human Multitasking

Thus far, detailed predictions of human performance using the Apex/CPM-GOMS architecture [1] have been confined to human-computer-interaction tasks. These models have emphasized the *execution-level* aspects of multitasking, by which we mean the manner in which efficient behavior is accomplished. Once it is decided which tasks to perform, the cognitive architecture determines the efficiency with which they are executed, i.e., their duration and resource requirements. The treatment of multitasking was restricted to parallel execution of operations from two successive actions. Indeed, a critical accomplishment was the development of a theory of the constraints on scheduling cognitive, perceptual, and motor resources. The scheduling of human resources (e.g., eyes, hands, speech) necessary to meet task demands is critical, as it underlies such phenomena as workload and throughput, and is important in understanding how efficiently two or more tasks can be done concurrently. In dynamic domains, such as air traffic control, other components of multitasking behavior emerge. The agent is embedded in a dynamically changing world, over which other agents also exert control. The agent must now have policies for servicing multiple tasks, including responding to unexpected events initiated by external agents, as in interruption [6,7]. This requires an extensive treatment of the *decision-level* aspects of multitasking, by which we mean the rules and knowledge that specify how the agent services the several tasks active at any moment. An extension of the Apex/CPM-GOMS compositional approach to handle the integration of *decision* and *execution* levels would represent a significant advance in the utility of the cognitive architecture.

Apex models an agent engaged in multiple tasks, deciding how to allocate limited resources to accomplish them. Thus, it includes support for modeling a wide range of multitasking behaviors common to complex domains [6,7]. The high-level architecture of Apex is shown in Figure 1. The *Resource Architecture* specifies the characteristics of the resources, including necessary preconditions, parameters for the time to complete actions, and the effect of the action. The *Action Selection Architecture* implements the policy for, and constraints on allocating resources, critical factors in determining how efficiently two tasks can be done concurrently. It assumes unary resources, but is otherwise neutral with respect to the resource model (it has even been used to model a simulated helicopter). It determines which tasks are active and how resources should be allocated. Tasks become active when the agent detects events that match the conditions on a procedure in Apex's *Procedure Library*. The *Action Selection Architecture* implements a reactive plan execution mechanism [6] that recursively decomposes high-level goals into subgoals,

creating a task hierarchy, consistent with most cognitive task analytic methods. The hierarchy is not generated all at once. Instead, the planner defers subgoal expansion until all preconditions are satisfied. Deferring goal decomposition until near execution time enables the planner to choose how to decompose (i.e. which procedure to use) with as much situation information as possible. This strategy is essential for tasks such as air traffic control where uncertainty about future world state can be high. The reactive planner also supports interruption by external events.

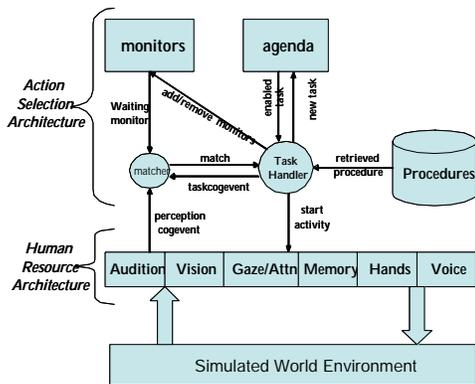


Figure 1. The Architecture of Apex

```
(procedure
(index (do-domain))
(step s (air traffic control))
(step t (end-of-simulation) (waitfor (end-sim-signal))))

(procedure
(index (air traffic control))
(step s1 (monitor radar) (priority 0))
(step s2 (initiate handoff for ?ac-symbol)
(waitfor (initiate-handoff ?ac-symbol))
(priority 4))
(step s3 (receive handoff request for ?ac-symbol)
(waitfor (receive-handoff ?ac-symbol))
(priority 2))
(step s4 (detect metering violation for ?ac-symbol)
(waitfor (detect-metering-violation ?ac-symbol))
(priority 6))
(step s5 (detect conflicts and resolution for ?ac-1 and ?ac-2)
(waitfor (detect-conflict ?ac-1 and ?ac-2))
(priority 8)))
```

Figure 2. Top Level Procedure of ATC

A Simple Apex Model of Handoff

Figure 2 shows the multitasking features of Apex used in two high level procedures of our air traffic control model. Every Apex procedure includes at least an *index* clause and one or more *step* clauses. The index identifies the procedure and specifies the class of goals for which it is appropriate. Each step clause describes a subgoal or auxiliary activity. Here the procedure "air traffic control" specifies five component controller activities. Steps are concurrently executable unless otherwise specified. In essence, at any moment Apex attempts to do as much as it can (in this respect it represents an upper bound on what humans are likely to do). Figure 2 shows two mechanisms for ordering the execution of steps, the *waitfor* clause and the *priority* clause. A *waitfor* clause is used to indicate preconditions, including ordering constraints. Goals created with waitfor conditions become enabled for execution only when all the events specified in the waitfor clause have occurred. Since *step s1 (monitor radar)* has no waitfor preconditions it is enabled immediately. The remaining steps begin in a *pending* state, becoming enabled

events match their preconditions. This procedure, then, describes the air traffic controller as continually monitoring the situation, suspending that activity when circumstances dictate one of the four specific activities indicated.

The *priority* clause enforces ordering when there is a conflict for a resource. For example, subgoals of *s1 (monitor radar)* and *s4 (detect metering violation)* involve eye fixations, which require the gaze resource. The metering subgoal has higher priority (6) than monitoring (0). Thus, when one of its subgoals requiring gaze becomes enabled it will seize the gaze resource, interrupting any active subgoal of monitoring requiring gaze, which will be suspended. Resumption of suspended tasks can occur in multiple ways once the high priority task is terminated. In the model described below, priority is the only basis for deciding which task to service. While overly simple, the examples below will show that it provides a good first approximation.

We have developed a simplified controller model focused on receiving and initiating handoffs to explore operator performance with new systems for automated handoff. The model predicts time and resource usage, both of which are necessary to provide insight into the mental demands placed on the controller in routine operations to permit estimates of workload, throughput, and suggest efficient ways to structure tasks. The agent communicates vocally with the pilot of each aircraft in its sector, and must wait to receive acknowledgement from the downstream controller when handing off an aircraft. Thus, we also provide simplified pilot and downstream controller agents. The sector controller model implements the flow chart of the handoff procedure shown in Figure 3, taken from existing task analyses [8,9]. To accept a handoff, the agent 1) detects the flashing aircraft symbol, 2) acquires the necessary situation awareness (3 sec), and 3) decides to accept (.5 sec), 4) executes acceptance by slewing the trackball cursor on the target and clicking the center button (time calculated by Fitt's Law). Durations were estimated from existing theory and analyses [8] and represent zero-parameter predictions of performance. It should be noted that the code in Figure 2 is sufficient to generate time estimates and switch between tasks based on priority. With additional Apex control statements the model could function as an agent in a dynamic simulation.

Figure 4a plots observed and predicted *execution* times for 10 of the 19 acceptances observed in an air traffic control simulation conducted by the Federal Aviation Administration. We derive the observed execution time by measuring the time from first fixating the aircraft to acceptance. The model prediction compares favorably to the observed times, with a Root Mean Square (RMS) error of 1.55 sec. Not shown are model predictions of parallel execution of monitoring and handoff subtasks. Consistent with those predictions, the data show instances of controllers communicating vocally with the pilot of an aircraft while sequentially fixating data blocks or aircraft symbols, or executing handoff acceptances from other aircraft.

The model's *decision* policy for accepting handoffs is very simple. Since accepting handoffs has a higher priority than monitoring, perception of the flashing icon causes the agent to suspend the monitoring task and begin the steps to accept the aircraft. We do not yet predict quantitative disengage and shift times. However, assumptions of the simple policy agree qualitatively with the total handoff times shown in Figure 4b, measured from the onset of flashing to the acceptance.

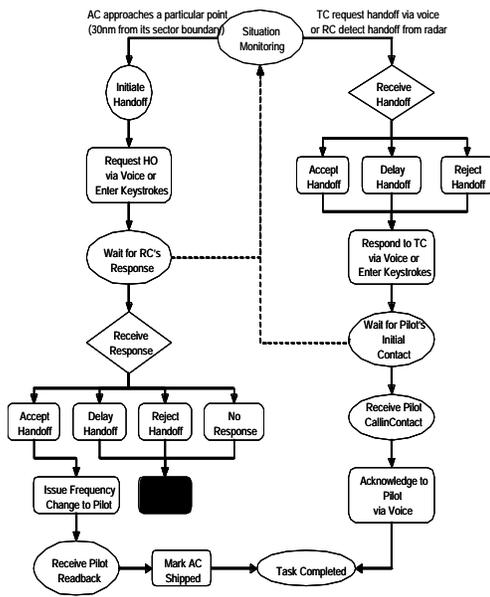


Figure 3. Task Analysis of a Handoff Task

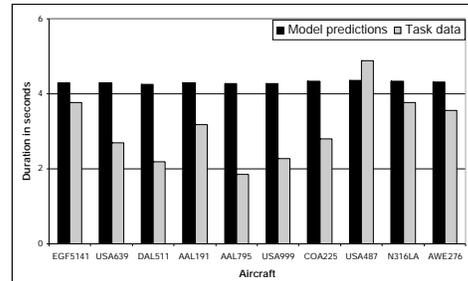


Figure 4-a. Comparison of Model Prediction and Task Data

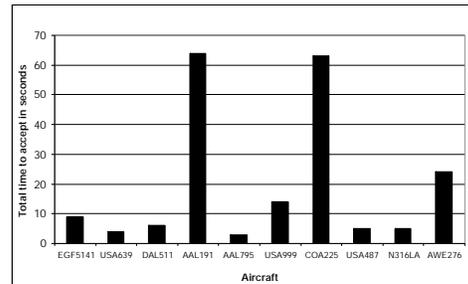


Figure 4-b. Total Time to Accept Handoffs

As Figure 4b shows, some of the acceptance times are quite long because of the delays in responding to flashing symbol despite fast execution once fixated. Thus, the variability is largely in suspending ongoing tasks to service the entering aircraft. Specifically, for the aircraft AAL191, USA999, COA225 and AWE276, when each of these symbols began flashing the controllers were engaged with higher priority tasks. For example in the case of AAL191 the R-side controller was aiding D-side controller to accept another aircraft while removing the interim altitude. In another case with COA225 their attention was drawn to a potential conflict which is given higher priority over accepting the aircraft. We expect the accepting handoff task would be suspended for higher priority tasks and resumed only after completing those tasks. In the case of AWE276 the controllers attended to a technical problem with a data link. Thus, cases of delay in responding to a handoff are consistent with priorities used by the model for selecting which task to undertake.

Conclusion

We have presented a framework for developing human performance models in complex domains and demonstrated performance predictions from a simple model of air traffic controllers accepting and initiating handoffs. Our results show that without estimating parameters from the task our Apex model was able to make accurate estimates of the time taken to execute the acceptance of a handoff. We have also shown how the design of Apex can support sophisticated policies for scheduling multiple concurrent tasks. It remains an important problem to characterize the policies used by experienced controllers in sequencing between concurrent tasks.

References

- 1 John, B. E., Vera, A. H., Matessa, M., Freed, M., and Remington, R. (2002): "Automating CPM-GOMS," in *Proceedings of CHI'02: Conference on Human Factors in Computing Systems*: New York, ACM Press, pp.147-154.
- 2 Freed, M., John, B., Matessa, M., Remington, R.W., & Vera, A. (2003): How Apex automates CPM-GOMS. In *Proceedings of the International Conference on Cognitive Modeling*, Bamberg, Germany.
- 3 Card, S. K., Moran, T.P. & Newell, A. (1983): *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- 4 John, B. E. (1996): TYPIST: A Theory of Performance In Skilled Typing. *Human-Computer Interaction*, 11 (4), pp.321-355.
- 5 Gray, W. D., John, B. E. & Atwood, M. E. (1993): Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance, *Human-Computer Interaction*, 8 (3), pp.237-309.
- 6 Freed, M. (1998): "*Simulating Human Performance in Complex, Dynamic Environments*," Doctoral dissertation, Northwestern University.
- 7 Freed, M. & Remington, R. (1997): Managing decision resources in plan execution. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*. Nagoya, Japan.
- 8 Leiden, K. (2000): "*Human Performance Modeling of En Route Controllers*," Micro Analysis & Design, Inc., Boulder, CO RTO-55 Final Report, Prepared for NASA Ames Research Center, December.
- 9 Niessen, C.; S. Leuchter; K. Eyferth. (1998): "A psychological model of air traffic control and its implementation," In *Proceedings of the Second European Conference on Cognitive Modeling*, Nottingham, U.K., pp.104-111.