Observations on Human Performance in Air Traffic Control Operations: Preliminaries to a Cognitive Model

Roger W. Remington
*Seung Man Lee**
*Ujwala Ravinder**
Michael Matessa
NASA Ames Research Center
*San Jose State University Foundation
MS 262-4 Moffett Field, CA 94035
650-604-6243, 650-604-4654, 650-604-6603, 650-604-6291
{rremington, smlee, uravinder, mmatessa}@mail.arc.nasa.gov

Keywords:
Air traffic control, human performance modeling, CPM-GOMS.

ABSTRACT: *We have previously described a computational approach to modeling human performance, built using the Apex architecture [1],in which extended behavioral sequences are automatically constructed from simpler CPM-GOMS templates [2]. Here we report on efforts to extend that method to the more complex domain of air traffic control operations. We describe the overall approach and our initial analysis of patterns of human performance observed in a simulation of air traffic control operations conducted by the Federal Aviation Administration. Of particular interest are patterns that characterize how operators manage multiple tasks, and distribute their attention over items. We show how templates in our compositional approach might be structured to accommodate these patterns.*

1. Introduction

Despite advances in automation, the human operator retains a pivotal role in challenging domains, such as air traffic control or air combat operations. The human operator brings a flexibility, adaptability, and resourcefulness unmatched by current automation. Unfortunately, human performance limitations are often a constraint on system performance, as well as a source of error. For example, the speaking rate of air traffic controllers can be the primary factor limiting throughput at peak periods for several hub airports, and memory failures have been implicated in airline accidents. Therefore, overall system performance must be considered with respect to human performance characteristics.

When designing new systems, or modifying existing systems, it would be useful then to be able to anticipate the response of the user before committing large resources in development. Our current methods for this have limitations. Even extended empirical user testing can fail to uncover serious difficulties. Such limitations have contributed to the interest in computational representations of the user that would allow the designer to simulate user responses to a variety of situations and design options. Though a complete model of the user is beyond current capability, several computational modeling approaches have been successful in making accurate predictions of user choices as well as task completion times [3-7]. Of the several architectures available to model human users, the Goals, Operators, Methods, and Selection (GOMS) method [3] has been the most widely used. A variant of GOMS, called CPM-GOMS has been shown to provide accurate, often zero-parameter, predictions of the routine performance of skilled users in procedural tasks [8].

In CPM-GOMS, as in the standard GOMS, a task is recursively decomposed from an initial Goal state into a nested set of subgoals. The user is assumed to have Methods that apply sequences of Operators that transition between subgoals to achieve the Goal. Selection rules are applied when there is more than one method to achieve a goal. In a standard GOMS analysis this task decomposition terminates in task-level primitives of any desired granularity. For example, a leaf node operator of a GOMS model of an air traffic controller might be to issue a clearance. In contrast, CPM-GOMS continues the decomposition, terminating in operators at the level of elementary human mental processes: Cognition, Perception, and Motor (hence "CPM"). One advantage of this is that leaf nodes at the

CPM level are constant across situations. Thus, it may be possible to achieve some domain generality by modeling at this level. Another advantage, which contributes significantly to the accuracy of prediction, is that this level of representation allows one to represent the parallelism that characterizes skilled behavior. In a standard GOMS representation, for example, performance times are computed by summing together individual leaf-node operator times. In CPM-GOMS, by contrast, Cognitive, Perceptual, and Motor processors can execute in parallel, which enables overlapping of the subcomponents of successive tasks.

1.1 Automating CPM-GOMS Modeling in Apex

A serious drawback to CPM-GOMS is the difficulty of constructing models at this level of detail. We have discussed this problem at length elsewhere and described success in automating the process using the Apex computational architecture [1, 2, 9]. Briefly, the implementation in Apex automates the sequencing of behavioral templates. Templates are psychological models of elementary behaviors in the task domain. For example, we have built templates for moving and clicking a mouse, for speaking, for typing, and other common behaviors. Each template consists of a sequence of Cognitive, Perceptual, and Motor operators that describe the putative internal operations required to perform the action. Operators in template are capable of executing in parallel subject to constraints. We have described elsewhere [9] how resource, logical, and slack-exclusion constraints are applied to determine the schedule of operations within a template.

By describing elementary behaviors common across task domains templates form an important bridge between the task domain and the underlying human performance architecture. Because templates represent activities common to many interfaces it should be possible to construct libraries of templates that can be applied in different circumstances. Because templates describe task-level activities it should be possible to model behavior without a deep understanding of cognitive psychology. That is, the psychology is embedded in the templates, and the description in the templates is sufficient for Apex, or other architectures to automatically interweave templates to form extended behavioral sequences. Indeed, the approach of composing behavior from template-level building blocks allows the modeler to use existing methods that analyze tasks by hierarchical decomposition, with templates as the leaf node operators.

1.2 Modeling Air Traffic Control

We have shown that this compositional approach can predict performance on routine, well-practiced tasks such as withdrawing a fixed amount of money from an automated teller machine, typing and mousing to interact with a computer-aided-design application, and in simple text editing. These are the domains in which CPM-GOMS has proven successful in past. In these tasks, users are highly practiced, and data is usually taken between 50 - 100 trials. At this level of practice the tasks are highly perceptual-motor in character and show the power of the CPM-GOMS method for capturing the peak performance of users.

The automation of CPM-GOMS in Apex for the first time allows us to address performance in more complex domains. We use complex loosely to describe a domain, such as air traffic control, where events are only quasi-predictable, people must juggle multiple tasks, and interruptions are present. Prior to the automation of CPM-GOMS it would have been far too labor intensive and error prone to begin to address domains such as air traffic control.

1.3 Extension to Complex Domains

A CPM-GOMS template provides a simplified characterization of human performance in terms of a schedule of limited resources. This fits well with Apex, which models an agent making decisions about how to allocate limited resources. A key question as we extend the approach to complex domains is whether we can retain the simplicity of this level of description, or whether it will be necessary to introduce more complicated architectural mechanisms to deal with knowledge and semantics. Behavior in domains such as air traffic control is characterized as a mixture of performance levels: skill, procedural, and knowledge. Because the description at the level of resources provides insight into issues such as workload and throughput, we wish to retain this essential character of CPM-GOMS. To extend the template approach it is necessary to identify those portions of the behavioral stream that contain routine behavior, and construct templates to model performance.

We have recently made progress on two major issues with respect to this extension. First, we have modified the Apex computational architecture to interact with external simulations. The advantage of this is that it allows us to leverage existing domain simulations, and provides an outlet for Apex models in the engineering community. Second, we have constructed templates for handoff operations based on task analyses. We have recently received data from simulations both of air traffic control and of shuttle cockpit operations. These data provide detail on external events, operator communications, operator actions, and operator eye

fixations at 60 Hz resolution. Below we describe how Apex interacts with external simulations, and describe our existing handoff model and compare it to data from the simulation. We also some features of the data that suggest ways in which to extend the template approach.

2. Integrating Apex with an External Simulation

Our air traffic control modeling supports the Virtual Airspace Modeling and Simulation (VAMS) project, a joint NASA and FAA effort. VAMS is developing the modeling and simulation infrastructure that would allow the effect of proposed changes to airspace operations to be evaluated early in the concept phase. The principal criteria for the evaluation are capacity (throughput) and safety. VAMS has developed a new airspace simulation system, ACES. ACES is a distributed agent-based event-driven simulation for the analysis of the NAS supporting common communications protocols. For the distributed simulation, ACES uses the High-Level Architecture (HLA) standard developed by the US Department of Defense (DOD) and the Run-Time Interface (RTI) available from the Defense Modeling and Simulation Office (DMSO). ACES provides key agent models for the NAS simulation, including aircraft, airport, Terminal Radar Approach Control (TRACON), Air Route Traffic Control Center (ARTCC), Airline Operations Center (AOC), and weather. These different types of agents are interacting with each other through message passing during the simulation. ACES models the dynamic behavior of different aircraft types, as well as the procedures and communications agents such as aircraft and air traffic control. In its current build, ACES represents agents at a level that abstracts over individual humans and human-system interfaces. Since an important goal for us is to evaluate such interfaces, we model the entire joint human-system, and use the combined output to control an ACES agent.

To introduce human behavior into the simulation we developed a simple Apex human agent model that simulates the functions of individual air traffic controller. One of challenges was to integrate the two different simulation systems to run seamlessly together. Apex is written in Lisp while ACES is a Java-based simulation system using HLA protocols. To enable communication between the two systems we developed communications interfaces and protocols using sockets. The Apex agent communicates with ACES agents through a TCP/IP socket channel as shown Figure 1. Conceptually, ACES is the direct link to all simulation objects. Apex interprets the simulation world and controls aircraft, as a human controller does. As a result, the Apex-ACES Interface (AAI) translates simulated displays and vehicle information into the representation used by Apex and translates Apex's action representation into ACES function calls (e.g., to control the vehicle and to manipulate sensors, radios, and displays). To control the vehicle's motion Apex issues output commands to a corresponding ACES agent. AAI converts the output commands to OpenCybele message format, which can then be interpreted by other agents in the simulation. Currently, control of a vehicle occurs through setting the desired state of the vehicle such as its speed, heading, and altitude.

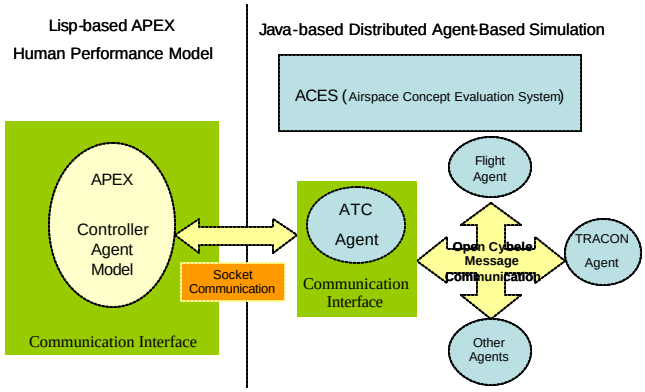

Figure 1. Integration of Apex into ACES

Apex is used to model both the human agent and the displays and controls the agent interacts with. In this way, timing of agent actions and delays imposed by the equipment and user interface can be simulated at high fidelity. Communication with the ACES simulation currently occurs at synchronized message passing times in accordance with ACES protocols. This arrangement is depicted in Figure 1. This scheme provides a natural division between the human performance model and the ACES-level agent. The ACES incorporates objects that represent high-level agents. The Apex agent need only transmit to its ACES counterpart those messages of significance in the larger context. Thus, while each keystroke of data entry into flight computers must be simulated to predict the time, the ACES-level agent need only be informed when the keystrokes produce some change in the its state, such as activating a mode, or changing a control setting. More timely communications could be achieved using asynchronous message passing. ACES does not support asynchronous timing mechanisms. We are working on establishing this capability.

3. A Simple Apex Model of Handoff

Our Apex agent performs the functions of an air traffic controller responsible for traffic in a single enroute sector during routine operations. In routine operations,

an enroute controller monitors the flights passing through the sector, responds to various pilot requests, and adjusts aircraft trajectories by instructing pilots to alter their aircraft speed, flight level, and heading in order to maintain safe and efficient flow of air traffic. Our goal is not to build an agent that correctly does air traffic control. Rather, it is to assess the demands of doing the work.

Previous work has attempted to provide a more or less complete functional analysis of enroute control [10-12]. A high level functional analysis of a portion of current enroute control is shown in Figure 2 Each sector is managed by two controllers, the radar controller (R-side) and the radar associate controller (D-side), whose duties complement and overlap. The R-side controller is the primary controller in contact with aircraft. The D-side controller assists, sharing responsibility for conflict detection and planning, as well as performing routine housekeeping.

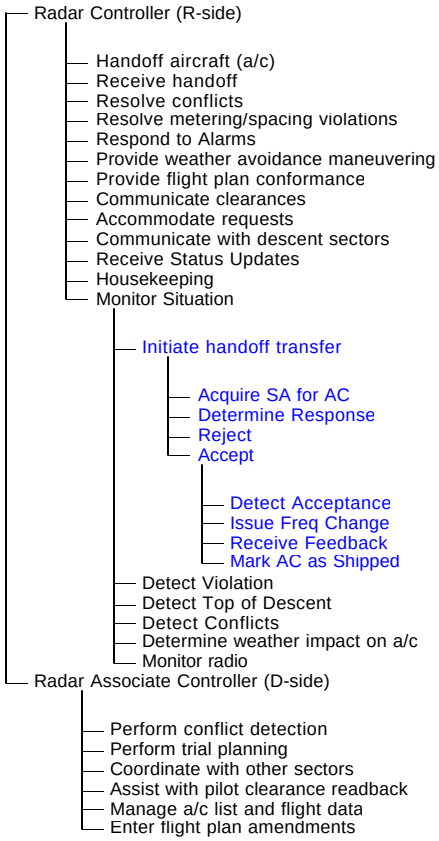

Figure 2. High-Level Tasks of Enroute Controllers

As shown in Figure 2, the air traffic controller must handle multiple tasks, one or more of which may be active at any moment. The multitasking behavior of our Agent model is constrained by policy, by data, and by resources. Policy determines which tasks should be given priority. This is governed by domain knowledge in combination with the current context. Within the Apex agent, tasks only become active when the conditions (data) are present to activate them. Resource constraints in the agent's architecture determine which task subelements can be done in parallel and which must wait for free resources. Tasks that require the same resources such as radio, trackball, or keyboard, can temporarily prevent parallel execution. In this way resources in our Apex controller agent capture capabilities and constraints on human behavior, which have consequences for overall performance.

Typically R-Side controllers initiate handoff when an aircraft reaches a trigger location (e.g. 30nm from boundary of next sector) and the receiving controller accepts a handoff before an aircraft enters his sector. Once a handoff is accepted, the transferring controller issues a frequency change clearance before the aircraft leaves his control sector. The details of this Transfer of Communications (TOC) differ for voice or datalink, and this difference is of interest in assessing the impact of datalink. Then flight crew will tune new frequency and make an initial contact to receiving controller in next sector. Figure 3 shows the high-level tasks of controllers for initiating and receiving handoffs.

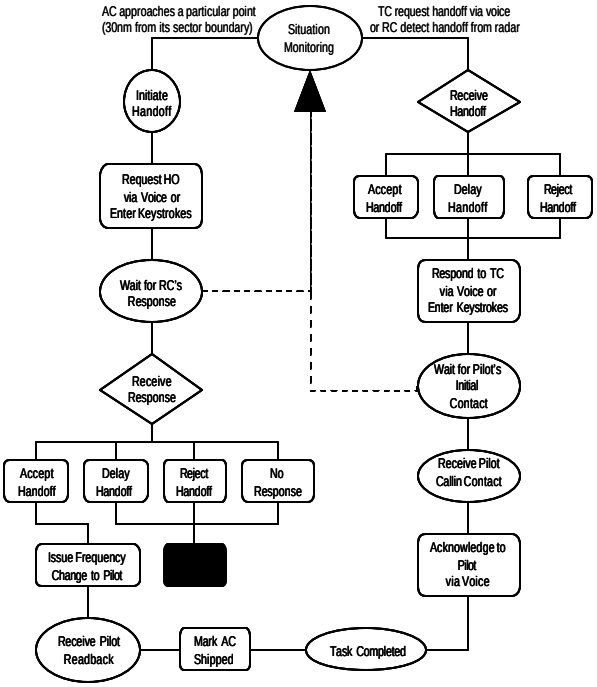

Figure 3. Task Analysis for a Handoff Task

These task analyses are represented in GOMS as a hierarchy of nested goals. This hierarchical structure is expressed in Apex using its Procedure Description Language (PDL). PDL steps are decomposed hierarchically into procedures of simpler steps until those steps bottom out in primitive actions that occupy

human resources. Figure 4 shows the top-level procedures. The ***index*** clause contains the name of the procedure and is used to locate the procedure in the procedure library. The ***air traffic control*** procedure consists of 5 steps, each of the procedures are executed when their conditions are met, as specified in the ***waitfor*** clause. In the case shown here, the 5 steps can execute in parallel and are assigned a priority, where the highest value has greater priority. Figures 5 & 6 show how the high-level PDL procedures for the handoff task that decompose successively until they bottom out into the primitive procedures shown in Figure 6.

```
(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))
          (period :recurrent :reftime enabled) (priority 4))
 (step s3 (receive handoff request for ?ac-symbol)
          (waitfor (receive-handoff ?ac-symbol))
          (period :recurrent :reftime enabled) (priority 2))
 (step s4 (detect metering violation for ?ac-symbol)
          (waitfor (detect-metering-violation ?ac-symbol))
          (period :recurrent :reftime enabled) (priority 6))
 (step s5 (detect conflicts and resolution for ?ac-1 and ?ac-
2)
          (waitfor (detect-conflict ?ac-1 and ?ac-2))
          (period :recurrent :reftime enabled) (priority 8)))
```

Figure 4. Top-Level Apex Procedures

```
(procedure
 (index (detect initiating handoff ?aircraft))
 (step s1 (decide whether to initiate handoff ?aircraft))
 (step s2 (initiate handoff ?aircraft to next controller)
          (waitfor ?s1))
 (step s3 (monitor response from receiving controller)
          (waitfor ?s2))
 (step s4 (issue frequency change to pilot) (waitfor ?s3))
 (step s5 (mark ac shipped)
          (waitfor ?s4 (pilot readback)))
 (step done  (terminate) (waitfor ?s5)))

(procedure
 (index (receive handoff request for ?ac-symbol))
 (step s1 (acquire sa for handoff ac))
 (step s2 (determine response) (waitfor ?s1))
 (step s3 (respond to initiating controller ?ac-symbol)
          (waitfor ?s2))
 (step s4 (wait for initial contact from pilot)
          (waitfor ?s3))
 (step done (terminate) (waitfor ?s4)))
```

Figure 5. High-Level Handoff Procedures

```
(procedure
 (index (initiate handoff ?ac-symbol to receiving
controller))
 (step s1 (move-cursor-to ?ac-symbol))
 (step s2 (click trackball on ?ac-symbol) (waitfor ?s1))
 (step done  (terminate) (waitfor ?s2)))

(procedure
 (index (move-cursor-to ?target))
 (profile right-hand)
 (step s1 (trackball-time ?target => ?time))
 (step s2 (start-activity right-hand moving-ic-trackball
                                  :object ?target :duration
?time => ?a)
      (waitfor ?s1))
 (step s3 (terminate) (waitfor (completed ?a))))

(procedure
 (index (click trackball on ?target))
 (profile right-hand)
 (step s1 (trackball-object => ?object))
 (step s2 (start-activity right-hand clicking-trackball
```

Figure 6. Primitive Procedures

4. Patterns of Controller Behavior

We have recently received data from an air traffic control simulations provided by the Federal Aviation Administration. Our initial analysis has focused on two areas that bear on the extension of our template approach. First we examine patterns of multitasking behavior to determine how to structure the procedures in the initial Apex model. Next we examine how the operator's focus of attention is distributed over the aircraft in the sector. This will begin to tell us behavior patterns in routine behavior, and how factors that govern the selection of which task to work.

4.1 Concurrent Task Management

In Figure 4 we presented an initial high-level Apex procedure consisting of several tasks. Our assumption was that these tasks would be executed concurrently with dynamic priorities set by policy and practice. The data provide some insight into how controllers in this simulation actually managed these multiple tasks.

Figure 7 plots the eye fixations and events associated with one specific aircraft, USA639, as a function of time in the simulation. Each of the data points represents a fixation on the aircraft or its data tag at that point in time, the vertical lines represent actions taken at specific times. Eye fixations for two other aircraft, BTA5093 and AAL904, are also plotted. The sequence of actions associated with USA693 illustrates the responsibilities and functions of the controller consistent with our initial model taken from the task analysis.

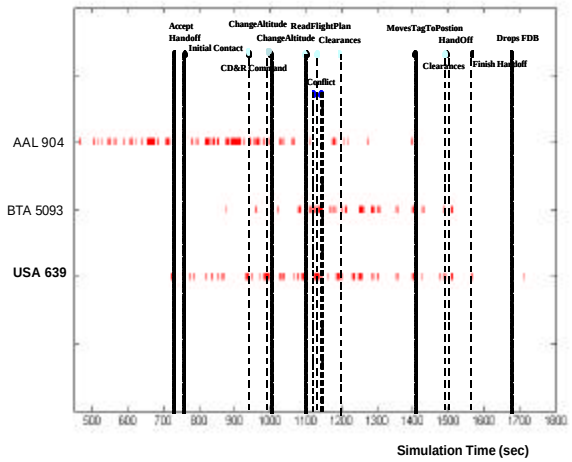

Figure 7. Eye Fixations and Actions for USA639

Because the time scale of 30 min is much greater than the typical fixation duration of about 300-500 ms, it may appear that at some points in time all three aircraft are being fixated simultaneously, when in reality controllers fixate aircraft serially. Serial behavior of this sort can still be considered multitasking. In some cases the individual behavioral elements can be related to multiple high-level tasks. For example, there is a conflict alert at about 1100 sec indicating a predicted conflict between USA6392 and BTA5093. The controller appears to be aware of that in advance given the high number of fixations on USA639 preceding the alert. Still, during that interval there are several fixations on AAL904, an aircraft not involved in the conflict, yet relatively few fixations on the other conflict aircraft BTA5093 until just prior to the alert. The fixations on AAL904 are likely related to a different set of high-level tasks than those associated with the conflict, and the controller shares monitoring resources with these ongoing tasks. The identification of multitasking cannot be made entirely based on concurrent execution but must consider the span of events or time represented in the working memory of the controller.

Some idea of the multitasking load can be seen in Figure 8, which plots the dwell times for each of a set of aircraft over a 5 min interval. Dwell times were computed by summing individual fixation durations on each aircraft over a 1 min span. If we assume that a fixation represents attentional resources being devoted to an aircraft, then over intervals of this duration operators attend to 8-15 aircraft.

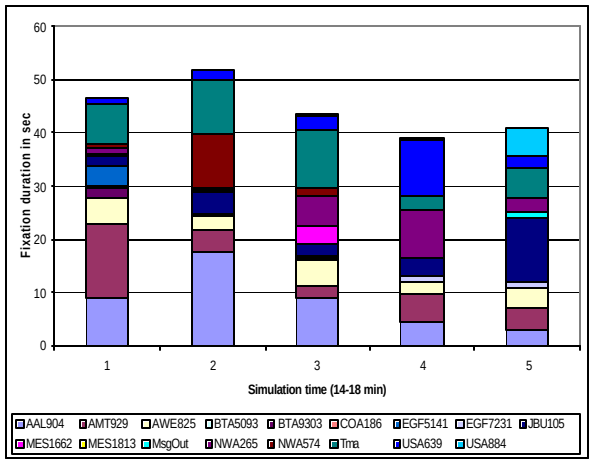

Figure 8. Dwell Times on Individual Aircraft

Is it possible to identify episodes of true concurrency? It would be expected that when controllers were required to wait that they would attend to multiple aircraft and perhaps service other tasks. Unfortunately, the data do not contain the events needed to obtain such estimates. However, there are several instances where controllers conduct monitoring scans while giving vocal clearances. This is shown in Figure 9. Here the controller stars the utterance and while speaking fixates several aircraft. We see clear evidence that monitoring is conducted in parallel with other behaviors, probably as a routine background activity.

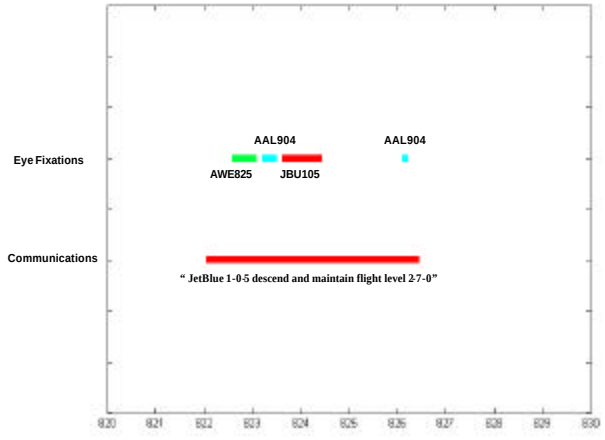

Figure 9. Concurrent Vocalization and Fixation

Examples such as the one above show that when resource and logical constraints allow, true concurrent performance will be observed. It also points out the need for a well-specified resource architecture at the level of elementary mental operations. Without a detailed consideration of the cognitive, perceptual, and motor level of performance such concurrency could not be accounted for.

4.2 The Focus of Attention

In addition to data on the sharing of attention, Figure 8 also shows how the focus of attention on any individual aircraft shifts as a function of time. For example, in the first minute two aircraft, AAL904 and AMT929, have long dwell times, which decrease over the five minute interval. We are currently working to relate these changes to specific operational situations or actions to determine patterns that exist that would inform us how to structure our domain-specific templates.

Another way to track the focus of attention is to see how concentrated or dispersed it is at any point in time. Figure 10 attempts to depict this by plotting three quantities as a function of time: the total number of aircraft, the number of different aircraft attended (fixated), and the total time spent looking at aircraft.

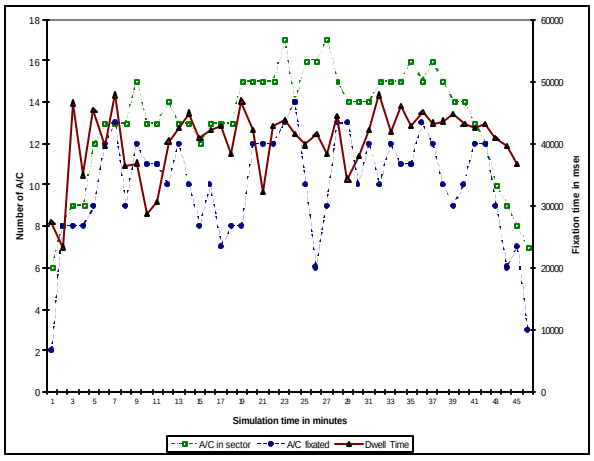

Figure 10. Fixation Patterns over Time

5. Summary

We have described some preliminary steps in generalizing a template-based approach to modeling to complex domains. The model developed largely from task analyses is now being modified in response to patterns of data observed in a simulation. The data contain important observations of how controllers share tasks and how their focus of attention is directed by contextual information. Many of these features needed to handle these examples of multitasking were already incorporated into Apex. The simulations provide us with the content needed to determine how to use the existing functionality. At present the only outstanding issue is how to dynamically adjust priorities. We are currently running simulations to compare the performance of our model to observed results and refining the model.

6. Acknowledgements

This work was supported by the Virtual Airspace Modeling and Simulation project of the NASA Airspace Systems program. The authors wish to thank Lisa Bjarke, Karlin Roth, and Larry Meyn for programmatic assistance, and Parimal Kopardekar, Ben Willems and Ken Leiden for making available task analysis and task modeling data.

6. References

[1] Freed, M. "Simulating Human Performance in Complex, Dynamic Environments," Doctoral dissertation, Northwestern University, 1998.

[2] John, B. E., Vera, A. H., Matessa, M., Freed, M., and Remington, R. "Automating CPM-GOMS," in *Proceedings of CHI'02: Conference on Human Factors in Computing Systems*. New York, ACM Press, pp. 147-154, 2002.

[3] Card, S. K., Moran, T.P. & Newell, A. *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates 1983.

[4] John, B. E. & John, B. E. & Kieras, D. E. "Using GOMS for user interface design and evaluation: Which technique?," *ACM Transactions on Computer-Human Interaction*, New York: ACM Press, 3(4), pp. 287-319, 1996.

[5] Kieras, D. E. "Guide to GOMS model usability evaluations using NGOMSL," *The Handbook of Human-Computer Interaction*, M. Helander and T.Landauer (Eds.), 2nd ed. North-Holland Amsterdam.

[6] Kitajima, M. & Polson, P. G. "A comprehension-based model of correct performance and errors in skilled, display-based, human-computer interaction," *International Journal of Human-Computer Studies*, 43(1), pp.65-99, 1995.

[7] Pirolli, P. and Card, S. K. "Information foraging," *Psychological Review*, 106, pp.643-675,1999.

[8] Gray, W. D., John, B. E. & Atwood, M. E. "Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance," *Human-Computer Interaction*, 8(3), pp.237-309, 1993.

[9] Matessa, M.; A. Vera; B. E. John; R. Remington; M. Freed. "Reusable Templates in Human Performance

Modeling," In *Proceedings of the Twenty-fourth Annual Conference of the Cognitive Science Society*, 2002.

[10] Seamster, T. L.; R. E. Redding; J. R. Cannon; J. M. Ryder; J. A. Purcell. "Cognitive task analysis of expertise in air traffic control," *International Journal of Aviation Psychology*, Vol. 3, pp. 257-283, 1993.

[11] Leiden, K. "Human Performance Modeling of En Route Controllers," Micro Analysis & Design, Inc., Boulder, CO RTO-55 Final Report, Prepared for NASA Ames Research Center, December 2000.

[12] Niessen, C.; S. Leuchter; K. Eyferth. "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, 1998.

Author Biographies

SEUNG MAN LEE

Seung Man Lee is a senior research engineer of Cognition Laboratory at NASA Ames Research Center. He received his Ph.D. in Industrial and Systems Engineering from Georgia Institute of Technology. His research interest includes the modeling and simulation of human behavior and cognition, and the development of human agent models for large-scale agent-based simulations.

MICHAEL MATESSA

Mike Matessa received his Ph.D. in Psychology from Carnegie Mellon University. He has had a decade of experience with the ACT-R architecture working with John Anderson at Carnegie Mellon University. He developed the Visual Interface, which was the first attempt to have ACT-R process perceptual information and perform motor actions. His current work if focused on the application of computational models of cognitive processing to aviation applications.

UJWALA RAVINDER

Ujwala Ravinder received her M.S. degree in Computer Science from University of Minnesota, Twin Cities. She is working as a senior research associate in the Cognition laboratory at the NASA Ames Research Center. Her current work is focused on task analysis of complex domains, developing and analyzing human performance models in the aviation domain.

ROGER REMINGTON

Roger Remington received his Ph.D. in Human Experimental Psychology from the University of Oregon in 1978. He is currently the Group Lead for Cognition at the NASA Ames Research Center. His research focuses on human visual attention and multitasking. His work has emphasized the application of computational models of human performance to aerospace domains.