Adding a Motor Control Component to the Operator Function Model Expert System to Investigate Air Traffic Management Concepts Using Simulation*

Sinem Göknur, Matthew Bolton, Ellen J. Bass
Department of Systems and Information Engineering
University of Virginia
Charlottesville, VA U.S.A.
scg5e, mlb4b, ellenbass@virginia.edu

Abstract – *This paper describes an enhancement of the Operator Function Model Expert System architecture that integrates automated agent control capabilities with an agent based simulation investigating Distributed Air/Ground Traffic Management. The enhanced architecture incorporates a new task manager module that handles task prioritization using a modified version of Cockpit Tasks Management, a strategies and operational heuristics module that provides methodologies for prioritizing tasks and selecting operator actions, a simulation control module that emulates user input, and an interface to the state variables of the controlled simulation. The paper also discusses a new data visualization and analysis tool for analyzing the performance of the automated agent.*

Keywords: Automated agent, distributed air-ground traffic management, human performance models, intelligent systems, operator function model, task management.

1 Introduction

NASA's Distributed Air/Ground Traffic Management (DAG-TM) program is developing a set of operational concepts, procedures, and decision support tools to aid in improving operations of the National Airspace System (NAS). Under many emerging concepts, pilots are expected to have additional responsibilities associated with maintaining aircraft separation [1], [10], [11]. The key component of NASA Langley Research Center's operational concept is that trained flight crews of properly equipped aircraft can assume full responsibility for separation from similarly equipped traffic in the enroute and terminal-transition domains while aircraft not in this category would continue to receive separation services from ground-based air traffic management. Shifting responsibility for ensuring traffic separation of equipped aircraft to the flight crew changes the separation assurance task for both pilots and air traffic controllers. That is, offloading air traffic controllers is accomplished by distributing the separation assurance task among multiple flight crews.

To aid flight crews, decision support tools and advanced cockpit displays are under development to provide automated conflict detection and resolution support including multilevel alerting [1], [2], [3], [14]. The principal component of this toolset is the Autonomous Operations Planner (AOP) [1]. Using ownship flight plan and state information, Automatic Dependent Surveillance – Broadcast (ADS-B) of position and intent information from other aircraft in proximity to ownship, ground-based air traffic constraints, and other data, AOP performs trajectory planning that accounts for conflicts with traffic hazards, aircraft performance limitations, traffic flow management constraints, airspace constraints including weather and special use airspace, and operator flight goals, such as efficiency and schedule.

AOP has a context sensitive alerting scheme. It provides alerts based on the time to the potential conflict and whether an aircraft has priority. AOP generates a level 0 alert if an aircraft is a possible, but not current, threat to the ownship. It generates a level 1 alert if a long-range conflict is predicted. Based on company policy, pilot action may be suggested but not required for a level 1 alert. AOP generates a level 2 alert, known as a conflict detection zone (CDZ) alert, if the conflict is predicted to occur such that the flight crew must take timely action to resolve the conflict. A level 3 alert known as a collision avoidance system (CAS) alert indicates that a collision may be impending within one minute as AOP predicts passing within 0.15 nautical miles and 300 feet of another aircraft. This would be the final alert before a Traffic Alert and Collision Avoidance System (TCAS) warning. A level 0 alert will not necessarily become a level 1 alert. However, a level 1 alert will upgrade to a level 2 if no one takes action. A level 2 alert may not become level 3 as the aircraft may not be on a collision course.

Studying the interaction between pilot behavior and this alerting automation is an important step in the design

* 0-7803-8566-7/04/$20.00 © 2004 IEEE.

of the alerting system interfaces and the associated procedures. NASA Langley already has a simulation capability where up to eight human pilots can fly simulated aircraft that are interacting as part of a larger traffic management simulation, where the simulated aircraft is known as the Aircraft Simulation for Traffic Operation Research (ASTOR) (e.g., [2]). However, using human subjects in all phases of development is expensive. In addition, if the operational concepts to be tested are novel, it may be difficult to train human operators to not be biased by current operations [4]. Having simulated pilots control aircraft can therefore aid in the operational concept development process. The idea is that each simulated pilot may behave based on individually designed models. Simulation testing with traffic and hazard scenarios can then highlight potential design issues.

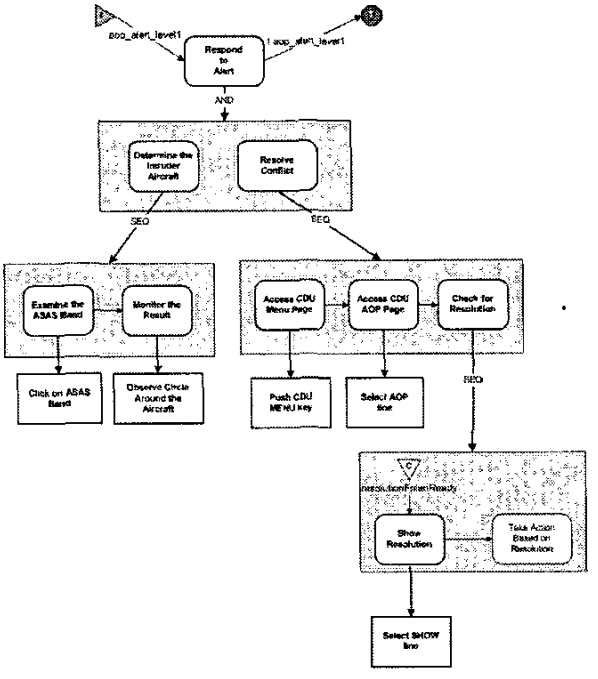

Figure 1. Operator Function Model for AOP Alert Level 1

Our new architecture allows a simulated pilot model to control an ASTOR aircraft. It uses the operator function model (OFM), a normative, heterarchic-hierarchic model that represents human operator activities in a dynamic, event driven world [8] [13]. Activities are described at various levels of abstraction: as major functions at the highest level, then subfunctions, tasks, and finally human operator (i.e., pilot) actions. Each OFM contains an initialization event and a termination condition. Figure 1 and Figure 2 depict an example OFM for responding to an AOP level 1 alert. The rounded-corner rectangles indicate functions and tasks, while operator actions are represented by square-cornered rectangles. Tasks may be composed of sub-tasks. When sub-tasks must be completed in order, the task includes a sequential (SEQ) indication. When sub-tasks must be completed in any order, the task includes an AND indication. An OR indicates that at least one of the tasks must be performed. An XOR indicates that only one of the tasks can be performed.

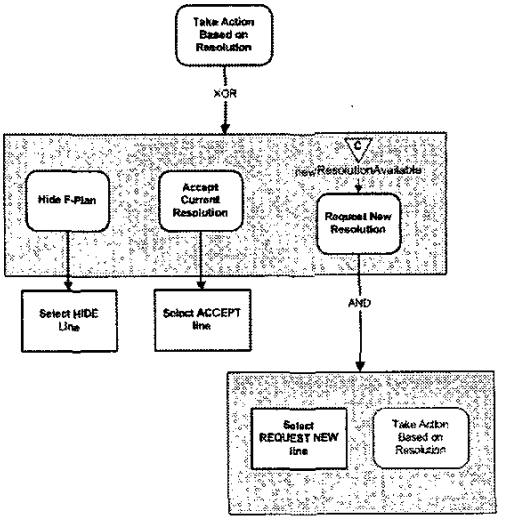

Figure 2. Operator Function Model for Take Action Based on Resolution

OFMspert is an object-oriented architecture which uses an OFM model to perform real-time intent inferencing of operator actions [13]. The existing domain-independent OFMspert instantiation, however, does not include a control component. Therefore, it cannot execute the actions specified in the OFM.

We are enhancing OFMspert with a control component. To accomplish this enhancement, OFMspert requires a means to control the simulation's interface, the ability to order AND sub-tasks, the ability to choose options for OR and XOR sub-tasks, knowledge and processing to assess task priorities, and the ability to interrupt the processing of a function, or task for one of higher priority. We are therefore making the following modifications to the OFMspert architecture (Figure 3):

- Simulation control executes the highest priority action;
- A new task manager module handles dynamic task prioritization;
- The strategies and operational heuristics module includes the schema necessary for prioritizing tasks and setting sub-task execution order; and
- Shared memory allows state data to be accessed by OFMspert.

In addition, we have also designed a data analysis tool which allows results from the simulated pilot model or from human pilots to be displayed and analyzed.

2 Control component

The control component is composed of four modules: simulation control, task manager, strategies and operational heuristics, and shared memory (Figure 3).

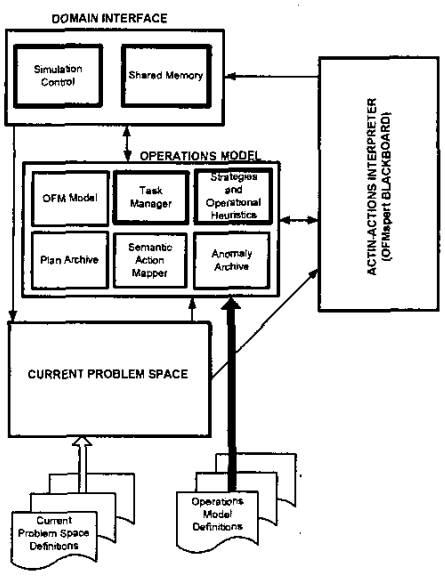

Figure 3. Modifications to the OFMspert Architecture

2.1 Simulation Control

To execute the current prioritized action, simulation control must be able to recognize and execute every action in the OFM. Simulation control has access to the simulation variables through the shared memory module. It also receives OFM state data from the OFMspert blackboard. The module controls the simulation by emulating user inputs such as engaging autoflight modes on the Mode Control Panel (MCP), selecting menu options on the Control Display Unit (CDU), and changing the range on the Navigation Display (ND). These actions are achieved via mouse movements, mouse clicks, and keystrokes. In our implementation, mouse movement, mouse clicks, and keystrokes are simulated using Windows API calls.

2.2 Task manager

In the case of concurrent tasks, the task manager constantly prioritizes the available tasks and sends the subsequent task execution information to the simulation control. The task manager accesses the blackboard in order to receive a list of all available tasks. It then prioritizes these tasks based on the priority information that is in the strategies and operations heuristics module.

This functionality draws from Cockpit Task Management (CTM) [5], [6], [7], a schema for handling the initiation, monitoring, prioritization, execution, and termination of multiple, concurrent tasks. Cockpit task management is a procedure executed by the flight crew in order to manage an agenda of flight tasks (Table 1; Figure 4). In the CTM jargon, an *event* is a set of system behaviors in which a defined change of state components has occurred. A *goal* is a set of desired behaviors that is achieved if all of the required behaviors have been executed properly by the system. An *initial event* defines the conditions required for a goal to become relevant. A *subgoal* contextually defines a portion of a goal. A goal can be viewed as a hierarchy of subgoals. *Priority* represents an ordering of goals according to relative importance. A *task* is a process that must be completed in order for a goal to be achieved. A task is initially *latent*. It becomes *pending* as its initial event approaches. A task becomes *active-not-in-progress* after its initial event has occurred. A task becomes *active-in-progress* if the resources it needs to achieve its goal have been allocated to it. A task returns to the *active-not-in-progress* state if these resources are deallocated from it. *Termination* occurs if the task's goal is achieved, if it cannot be achieved, or if it becomes irrelevant. An *agenda* is a hierarchy of tasks to be completed to achieve a higher objective. *Concurrent tasks* are tasks that are active (either active-not-in-progress or active-in-progress) at the same time. *Resources* are components of the domain that tasks need to manipulate. A *resource conflict* occurs when two or more concurrent tasks require resource beyond the capacity of the domain.

Table 1. Cockpit Task Management procedure [6], [7]

1. Create initial agenda
2. Until mission goal is achieved or determined to be unachievable:
   a. Assess current situation.
   b. Activate tasks whose initial events have occurred.
   c. Assess status of active tasks.
   d. Terminate tasks with achieved or unachievable goals.
   e. Assess task resource requirements.
   f. Prioritize active tasks.
   g. Allocate resources to tasks in order of priority:
      i. Initiate newly activated high-priority tasks.
      ii. Interrupt low-priority tasks (if necessary).
      iii. Resume interrupted tasks (when possible).
   h. Update agenda.

OFMs are particularly well suited to CTM for several reasons. Firstly, the set of all OFMs for a domain can be viewed as an agenda, with each individual OFM representing a particular high level task in the agenda. A task itself can be viewed as a hierarchy of subgoals or subtasks that collectively achieve the OFM's goal.

Because CTM is being incorporated into OFMspert, some changes are required. The primary difference between the original form of CTM and this modified version relates to presence of the blackboard in OFMspert (Figure 5). Because the blackboard automatically keeps track of what initialization events have occurred and terminated, the CTM need only access this information from the blackboard and assess the priorities of the tasks it depicts. Also task prioritization knowledge is drawn from the strategies and operational heuristics data and shared memory data from the modified OFMspert model.

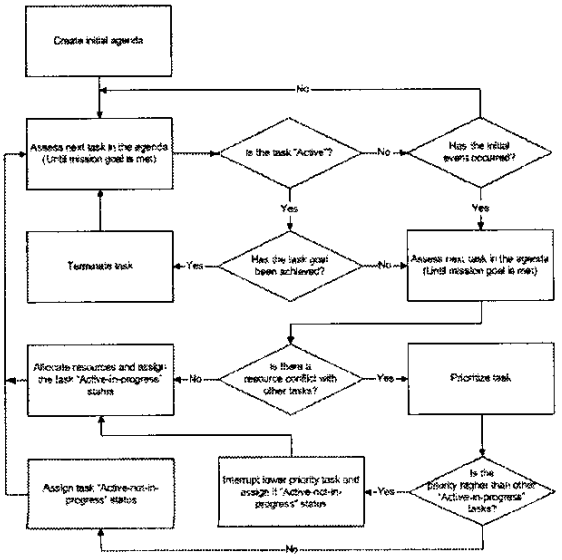

Figure 4. Cockpit Task Management algorithm [5]

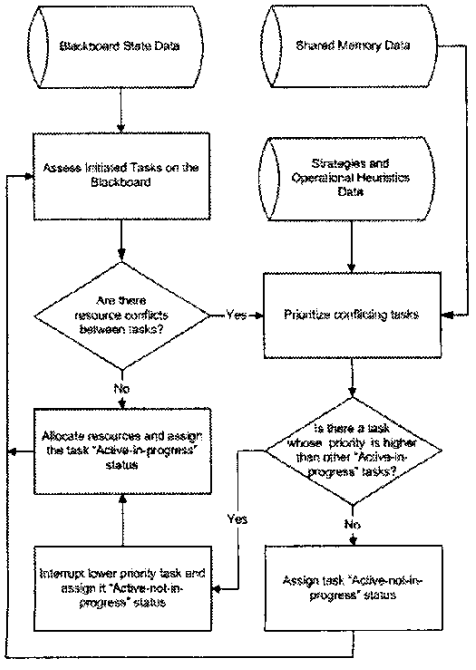

Figure 5. Modified Cognitive Task Management algorithm

It is important to note that this CTM is a continuous process in which active tasks are continually analyzed thus allowing prioritization schemes to be as flexible as possible. This will be discussed further in the next section.

2.3 Strategies and operational heuristics

The Strategies and Operational Heuristics module provides two important types of information. Firstly, it provides the task manager with the information it needs to prioritize tasks. Because the task manager has access to information from the shared memory and the blackboard, the prioritization scheme can be defined using variables from either of these sources. This can make the priority rankings of tasks extremely dynamic as they could be dependent on the amount of time tasks have been in "Active-not-in-progress" status, the amount of time tasks have been in "Active-in-progress" status, or how close the task is to being completed. Because task prioritization is dynamic, it should compensate for what Funk calls Task Status Assessment [6] [7], where a task's priority would decrease relative to other concurrent tasks as it becomes less likely that its goal will be achieved.

For responding to AOP alerts, the task prioritization scheme is relatively simple. Each OFM is initiated by the presence of an alert. Because these alerts are already prioritized based on the time to a loss of separation, Level 3 alerts have the highest priority, then Level 2, then Level 1, and finally Level 0. In situations where there are multiple alerts of the same level, priority is given to the alert that has the least time to loss of separation.

The second type of information provided by Strategies and Operational Heuristics module relates to how the simulation control executes SEQ, AND, OR, and XOR sub-tasks. The module provides a selection function for each sub-task in each OFM. Like the task prioritization scheme, a selection can be a product of a variable on the blackboard or shared memory thus supporting context sensitivity. Selection functions can also be based on random variables with defined distributions.

Multiple instances of a single activity can appear on the blackboard. For example, multiple ASAS bands (defining heading, indicated airspeed (IAS), and vertical speed values leading to a conflict) can be displayed simultaneously on the ND when the control component should click on an ASAS band. In these situations, the Strategies and Operational Heuristics module must provide a means of resolution. For the given example, this could take the form of progressively examining every ASAS band based on how close it is to the aircraft's current heading, IAS, or vertical speed values.

2.4 Shared Memory

Because OFMspert is separate from the simulation with which it is interacting, it needs a means of monitoring the simulation's state data. This is provided by the shared

memory module by means of a shared memory buffer called the Avionics Bus. The Avionics Bus simulates an ARINC 429 data bus by providing multiple channels of communication between the different modules that compose ASTOR [12].

Since all information relevant to ASTOR can be retrieved through the Avionics Bus, this is the ideal means for OFMspert interaction. However, problems arise due to the fact that OFMspert is written in Java and ASTOR is written in C++. In order to circumvent this problem, the shared memory portion of OFMspert utilizes the Java Native Interface (JNI). The JNI is an interface provided by the Java Development Kit that allows the Java virtual machine to interact with native (platform dependent) methods. This allows OFMspert to send messages to and receive messages from ASTOR through a JNI implemented wrapper. This wrapper provides conversions between C++ and Java data types and converts OFMspert requests into Avionics Bus messages.

3 Data analysis tool

The data output by ASTOR is a single file which reports every inter- and extra-process communication that occurs in a given simulation run. In this format, it is nearly impossible to interpret the data. In order to make analysis easier, the ASTOR software developers provide a set of Perl scripts. The Perl scripts parse ASTOR output data and organize the data into separate files by category. The scripts are accompanied by a text file describing what data are contained in each file and how coded variables should be interpreted.

In order to analyze the performance of the simulated pilot, our data analysis tool uses this parsed data to report when alerts occurred, when responses occurred, what actions the pilot has taken in response to alerts, how the aircraft state was affected by pilot action, and how well the pilot conformed to the AOP-predicted optimal state.

The analysis translates this data into a tailorable visualization (Figure 6). The most real-estate intensive portion is the rectangular plotting area. This area is composed of eight different plots, each sharing a common time-delimited horizontal axis. To enhance visibility, the time axis itself is displayed at both the top and bottom of the plotting area. The time domain can be scaled by using the box in the upper right side of the design. This box allows time ranges to be entered manually or scrolled through using up-down boxes. The selected time range can be set using the Set Time Domain button.

The data plotted in the display is intentionally ordered to match the stimulus and response nature of the pilot's interactions. The Alerts plots represent the stimuli; the ND, MCDU, and MCP plots represent the pilot's response; the Altitude, Heading, Indicated Airspeed, and Vertical Speed plots depict the resulting changes to the state of the aircraft.

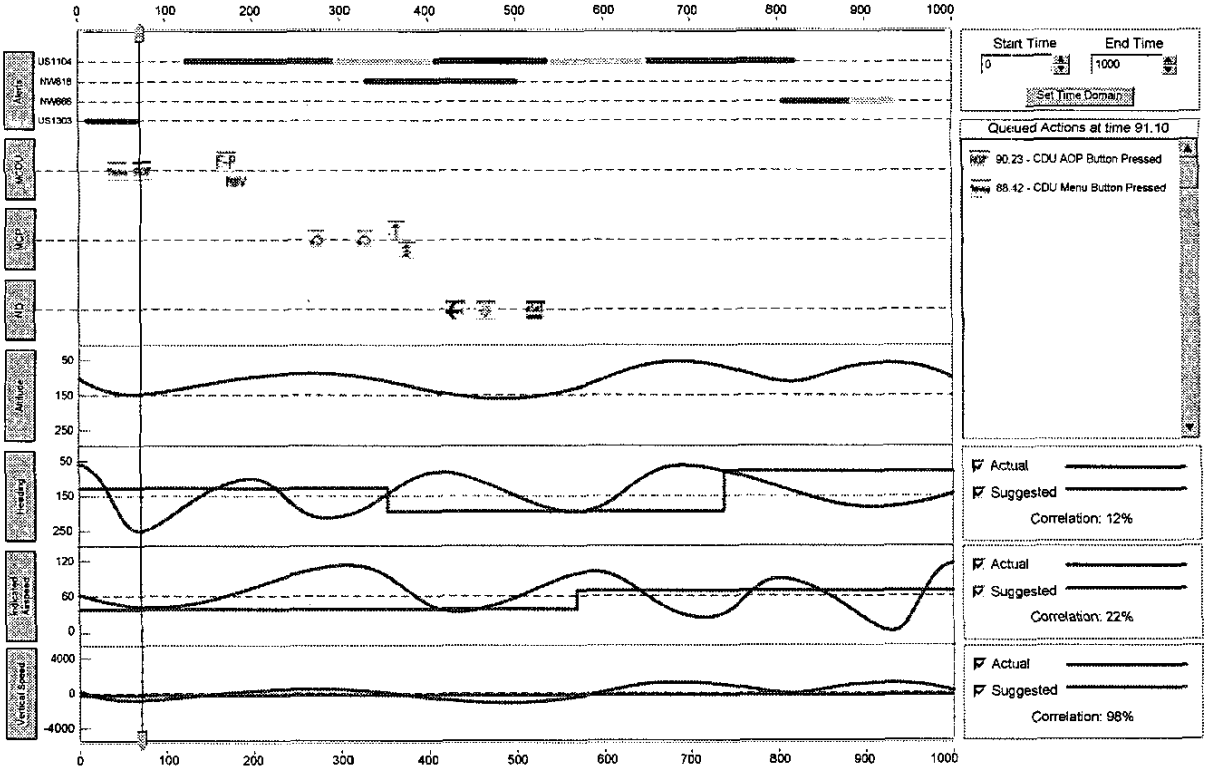

Figure 6. Data analysis tool

The latter four plots describe how a pilot's actions affected the state of the airplane (represented as altitude, heading, indicated airspeed, and vertical speed). Each of these plots contains a vertical axis on the left hand side of the plot area. A horizontal dotted line bisects each plot horizontally and represents the mid-point of the specified range. In each of these plots, the blue line indicates the actual state of the aircraft. In all but the altitude plot, a pink line indicates the AOP predicted optimal value. The degree with which the actual values correlate with those specified by AOP is reported in boxes to the right of the plots, one box for each of the three plots. The user can make either of these lines visible or invisible by clicking on checkboxes within them.

The Alerts section displays the alerts other aircraft caused with ownship. The Alerts section automatically scales itself so that it displays all aircraft that caused an alert for the set time domain. Each aircraft is given its own plot line. Alerts are represented by lines color coded to represent their alert level (blue = level 1, yellow = level 2, red = level 3). The span of the band indicates the time period for which the alert was valid. A black band represents a loss of separation.

The ND, MCDU, and MCP plots depict actions that occurred through the navigation display, multi-function control display unit, and mode control panel respectively. These are intentionally ordered to reflect the sequence of actions a pilot performs when responding to an alert:

- The pilot gains more information about an alert through interaction with the ND.
- The pilot tries to select a strategic solution through the MCDU.
- If a strategic solution is not available, the pilot attempts a tactical solution through the MCP.

The actual plot area for each of these three categories features a series of icons plotted along a central line. Each of these icons represents a user action that occurred. The leftmost position of the icon indicates when the represented action occurred. Each user action type is assigned a unique icon and color, with each icon designed around a consistent theme. Icons are designed to represent the actions they describe. For example, a pilot specified increase in altitude is represented by an icon depicting an up arrow. If icons overlap, they are staggered around the center line, ensuring that no user input display is obstructed.

An analyst wishing to gain more information about a particular pilot action has two options. He can either let his mouse hover over an icon and receive a tool tip description or he can use the action queue. The action queue is located directly below the time domain box. The action queue displays a list of all the pilot actions that occurred until the time indicated by the vertical time indicator line. This allows an analyst to get an accurate report of all the pilot actions that have occurred up to a specific time. An action's entry in the queue contains a description of the action along with its corresponding icon.

The vertical time indicator is a line that spans the vertical length of the plot area. It can be moved left or right in order to set an upper bound on the actions reported in the action queue. It can also be moved to align pilot actions with changes in the airplane state.

4 Conclusions

By coupling the development of the control component with the development of the data analysis tool, this research is providing a powerful means for developing normative operator function models and analyzing interfaces for the ASTOR simulation environment. While the control component allows for scenarios to be run without the need for human intervention, the data analysis tool gives researchers the means to determine how well the OFMs performed their tasks in those tests.

This paper has outlined a high level interpretation of how an OFMspert control component for an agent based simulation will function, but there is still work to be done. While the implementations for the Simulation Control and Shared memory components will be highly coupled to the domain being developed for, careful work will need to be taken to ensure that the Task Manger and Strategies and Operational Heuristics modules are not. This will be particularly challenging for the Strategies and Operational Heuristics module because it will require the development of functions that will be able to identify potentially complicated prioritization and execution ordering schemes based on domain-specific variables that can be easily transmitted to modules that need them. Future research will address these issues.

Acknowledgment

This work was supported in part by award UVA-03-01, sub-award 3035-UV from the National Institute of Aerospace and by NASA Langley Research Center grant NAG1-02113 (Michael Palmer, technical monitor).

References

[1] M. G. Ballin, V. Sharma, R. A. Vivona, E. J. Johnson, & E. Ramiscal, "A flight deck decision support tool for autonomous airborne operations", AIAA-2002-4554, August 2002.

[2] R. Barhydt, T.M. Eischeid, M.T. Palmer, & D. J. Wing, "Regaining lost separation in a piloted simulation of autonomous aircraft operations", Proc. 5th USA/Europe Air Traffic Management R&D Seminar, Budapest, June 2003.

[3] B. Barmore, E. J. Johnson, D. J. Wing & R. Barhydt, "Airborne conflict management within confined airspace in a piloted simulation of DAG-TM autonomous aircraft operations", Proc. 5th USA/Europe Air Traffic Management R&D Seminar, Budapest, June 2003.

[4] T. Callantine, "Agents for analysis and design of complex systems", Proc. IEEE International Conference

on Systems, Man and Cybernetics, Tucson, pp. 567-573, Oct. 2001.

[5] K. W. Colvin, "Factors that Affect Task Prioritization on the Flight Deck", *Dissertation Abstracts International: Section B: the Sciences & Engineering*, Vol. 61(2-B), Aug. 2000.

[6] K. Funk, "Cockpit Task Management: A Preliminary, Normative Theory", *NASA Center for AeroSpace Information (CASI)*, 1991.

[7] K. Funk, "Cockpit Task Management: Preliminary Definitions, Normative Theory, Error Taxonomy, and Design Recommendations", *The International Journal of Aviation Psychology*, Vol. 1(4), pp. 271-285, 1991.

[8] C. M. Mitchell, "GT-MSOCC: A domain for research on human-computer interaction and decision aiding in supervisory control systems", Proc. IEEE International Conference on Systems, Man and Cybernetics, SMC-17, pp. 553-572, 1987.

[9] C. M. Mitchell, D. A. Thurman, D. M. Brann, A. R. Chappell, "OFMspert I: Operations Automation", Proc. IEEE International Conference on Systems, Man and Cybernetics, Piscataway, pp. 1152-1157, 2000.

[10] S. Mondoloni, M.T. Palmer & D. J. Wing, "Development of a prototype airborne conflict detection and resolution simulation capability." AIAA-2002-4446.

[11] NASA, "NASA Advanced Air Transportation Technologies Project: DAG-TM Concept Element 5 En Route Free Maneuvering Operational Concept Description", Contract NAS2-98005 Research Task Order 72, September 2002.

[12] M.T. Palmer & Ballin, M.G., "A high-performance simulated on-board avionics architecture to support traffic operations research", Proc. AIAA Modeling and Simulation Technologies Conference and Exhibit, 2003.

[13] D.A. Thurman, A.R. Chappell, & C.M. Mitchell, "An enhanced architecture for OFMspert: A domain-independent system for intent inferencing", Proc. IEEE International Conference on Systems, Man and Cybernetics, San Diego, pp. 955-960, 1998.

[14] D. J. Wing, K. Krishnamurthy & R. Barhydt, & B. Barmore, "Pilot interactions in an over-constrained conflict scenario as studied in a piloted simulation of autonomous aircraft operations", Proc. 5th USA/Europe Air Traffic Management R&D Seminar, Budapest, June 2003.