Concept Investigation via Air-Ground Simulation with Embedded Agents

Todd J. Callantine,[*] Jeff Homola,[†] JoeyMercer,[†] and Thomas Prevot[*]
San Jose State University/NASA Ames Research Center, Moffett Field, CA, 94035

and

Everett A. Palmer[‡]
NASA Ames Research Center, Moffett Field, CA, 94035

Next Generation Air Transportation System (NGATS) concepts require design and analysis of interactions involving human operators and new automation tools. Research is exploring how air-ground simulations with embedded agents can complement human-in-the-loop simulations for this purpose. This paper describes research toward integrating computational air traffic controller agents into a large-scale distributed simulation in the Airspace Operations Laboratory (AOL) at NASA Ames Research Center. Before detailing the agent model and architecture and discussing simulation integration issues, the paper describes a methodology for using agents to understand human-system integration issues and illustrates how the methodology can be applied to an example concept.

I. Introduction

Concepts of operation for The Next Generation Air Transportation System (NGATS) are currently taking shape. A common theme is the integration of advanced technologies to solve the capacity and efficiency problems of today's air traffic management (ATM) system without compromising its demonstrated high level of safety. NGATS concepts, and evolutionary pathways to them, need to be analyzed while still in the early stages of development. When the technological components of a concept are considered in isolation, certain assumptions about their operation make the concept amenable to analytical analysis.[1] Analytical studies can produce useful capacity, efficiency, and safety metrics, but future concepts also envision critical (albeit, in some cases, backup) roles and responsibilities for human operators. Information about the potential workload of human operators, effectiveness of required coordination, overall operational acceptability, coping strategies, and the acceptability and use of new technologies is crucial for human-systems integration. The principal method for studying human factors of proposed concepts is to develop prototypes of new technologies and test them in large-scale human-in-the-loop (HITL) simulations. However, HITL simulations can be expensive and time-consuming to conduct.

Another approach, prevalent in NGATS research plans,[2] entails using fast-time simulations with embedded autonomous agents. This research focuses specifically on agents that represent individual human operators providing air traffic control (ATC) services. Fast-time simulations with ATC agents have the potential to accelerate the pace of iterative HITL concept development by enabling different automation tool configurations and operator roles and responsibilities to be analyzed inexpensively. However, simulations with ATC agents produce fundamentally different results from those obtained in HITL simulations. As computational entities, ATC agents excel at performing tasks consistently. This quality can be leveraged to distinguish situations in which using specific strategies, methods, and supporting technologies are likely to be successful from situations in which they are not.

Previous agent-based simulation studies have produced interesting results using very simple agents and systematically varying only technological and environmental factors (e.g. automation configuration, aircraft equipage, traffic characteristics, winds, routes).[3] This research seeks to enrich the capabilities of ATC agents while staying true to this general methodology. A second aim is to produce data from ATC agents that can be compared to data from HITL studies. A straightforward way to ensure consistency is to conduct both HITL and agent-based

[*] Senior Research Engineer, Human Systems Integration Division, NASA Ames Mail Stop 262-4, AIAA Member.
[†] Masters Student, Human Systems Integration Division, NASA Ames Mail Stop 262-4.
[‡]Human Factors Researcher, Human Systems Integration Division, NASA Ames Mail Stop 262-4.

simulations within the same simulation environment. This way, researchers can verify effects observed with agents in other simulation environments, and present human subjects with the same conditions. Modified operator roles and responsibilities designed to address issues identified in HITL simulations can also be evaluated consistently using agents.

A simulation that can be used to realize these objectives is the large-scale distributed simulation in the Airspace Operations Laboratory (AOL) at NASA Ames Research Center.[4] The AOL hosts a simulation constructed from interconnected Multi Aircraft Control System (MACS) stations. It has been used successfully for a number of HITL concept studies.[5,6] Some autonomous agents have been integrated in MACS to perform aircraft control and other functions. ATC agents are currently implemented within the Trajectory-Centered Simulator (TCSim).[7] Under the proposed scheme, the agents will be implemented in MACS to 'staff' MACS ATC workstations normally staffed by human operators. This will enable closed-loop agent-based simulations with technology prototypes and aircraft performance consistent with HITL simulations, in addition to enabling mixed human and agent configurations that reduce required staffing levels and support part-task research.[8] Moreover, the capability to validate interesting results in the AOL will improve the utility of agent-based simulations in TCSim.

This paper begins with a general discussion of an agent-based analysis methodology and requirements for agents to support it. It then introduces an NGATS-relevant concept of operations in which advanced scheduling and trajectory-planning tools and delegation of airborne spacing and merging tasks to flight crews figure prominently. After describing areas for analysis, the paper describes a model and architecture for ATC agents. It then discusses agent integration with MACS, including how agents can use MACS prototype automation and other functionality. The paper concludes with a discussion of issues for further research.

II. Methodology and Requirements for Agent-based Concept Investigation

Computational agent models have different forms and different emphases, in terms of the human behaviors they simulate. This research extends efforts to develop *worksystem* models of radar ('R-side') air traffic controllers in which a single agent represents a single controller. The term worksystem distinguishes these agents as using representations tied to the ATM domain[9] to perform processes hypothesized to support roles and responsibilities of the human operator in current and future ATM systems. These ATC agents may be considered 'computational human performance models,' but they are distinct from cognitive human performance models that explicitly represent low-level cognitive processes.[10] They may also be termed *task-analytic*, because they use explicit computational models of tasks and the operational contexts in which controllers should nominally perform them.

Simulations with task-analytic worksystem models of ATC practitioners excel at identifying when normative task performance succeeds or fails. This methodology for agent-based concept investigation seeks to eliminate as much internal complexity as possible from ATC agents, and make interactions with the environment and other agents the principal focus of analysis. Once these interactions are understood, adding probabilistic behavior can highlight interesting safety-related effects of unusually long task-completion times or the random application of a control method—things that certainly occur in practice—but the methodology first seeks to provide a detailed understanding of how environmental factors and supporting technologies impact nominal task performance.

Simulations should proceed in a stage-wise fashion, through a series of trials in which technological and environmental factors serve as independent variables that drive variations in ATC agent performance. Key factors include aircraft equipage, traffic characteristics, winds, weather, and routes. Simulations should be performed for relevant ATC automation tool configurations and nominal agent task specifications for scenarios that represent each variation. TCSim includes mechanisms to systematically vary certain aspects of traffic scenarios automatically. This process can identify factors agent performance is sensitive to, setting the stage for a new round of simulations with a narrower scope. Multiple iterations of fast-time simulations with configurations can be conducted in parallel. Once key interactions and operational contexts are identified, these can be simulated in the HITL simulation environment, first with agents to validate prior results, then with human subjects.

From this methodology follow some requirements for agents. First, agent task specifications should represent operations specific to particular airspace sectors. Roles and responsibilities differ, for example, between en route and terminal-area air traffic controllers. Second, agents should be capable of applying various *control strategies*. The presence or absence certain technologies can affect the priorities controllers assign to different *control methods* (i.e. clearance types). Effects of technologies such as automation tools are reflected in the way an agent executes a particular control method. Control strategies specify control method priorities according to characteristics of particular spacing or separation problems. Salient characteristics include the types of aircraft involved, their category (i.e. departure, arrival, or over-flight), their equipage (e.g. data link, ADS-B, airborne spacing assistance), and other factors (e.g. time to sector exit). Third, agent performance should reflect the effects of multi-tasking. Task

management priorities, or *agenda formulation strategies,* determine the order in which actions needed in the immediate future should be performed. Like control strategies, agenda formulation strategies may differ according to airspace sectors and the specific roles assigned to controllers. Fourth, in early stages of analysis, agents should take nominal amounts of time to perform activities. Agents should also have the capability to sample activity times from appropriate distributions in later stages of analysis. Finally, agents should produce output that enables interactions to be analyzed and understood. Important contextual elements should be included in the data.

The following section presents a concept of operations slated for exploration in both HITL simulations and simulations with embedded agents. The section that follows describes areas for agent-based analysis.

III. Example Concept of Operations

A concept, called 'trajectory-oriented operations with limited delegation,' has qualities of future ATM concepts being developed for the NGATS. [11] The idea behind the concept is to achieve overall capacity and efficiency gains at acceptable levels of safety by integrating three facets of operations:

1) Time-based flow management to regulate traffic density;
2) Trajectory-based operations to create efficient, nominally conflict-free trajectories that conform to traffic management constraints;
3) Airborne separation assistance to maintain local spacing between aircraft.

Since its initial publication, enabling arrival aircraft to fly efficient Continuous Descent Approaches (CDAs) to their assigned runways has become a primary goal for the concept. It has also been extended to include an important 'arrival flow conditioning' role for participating airline operations centers (AOCs). Participating AOCs use schedule information to issue en route speed clearances to sequence and space arriving company aircraft with the goal of enabling them to fly uninterrupted CDAs. Controllers are responsible for adjusting the sequencing and spacing of aircraft from non-participating airlines, and for preventing crossing traffic from interfering with aircraft flying CDAs.

HITL explorations will examine the concept in both near-term and farther-term instantiations. In the near-term condition, controllers have only current-day information. In the farther-term conditions, runway schedules are considered to be shared among controllers and participating AOCs. To support the farther-term concept, the following prototype decision support tools have been developed in MACS: [5]

1) Timelines that display runway schedule information;
2) Trajectory-based trial-planning tools for graphically creating conflict-free trajectories that meet schedule constraints;
3) Spacing advisories that indicate lead aircraft assignment, and current and advised spacing;[6]
4) Integrated data link that enables controllers to uplink Flight Management System (FMS)-loadable trajectory clearances and spacing information to aircraft.

The concept has been instantiated for the airspace shown in Figure 1. Four airspace sectors are considered: a terminal area, two high altitude sectors, and a low altitude sector that lies beneath the high altitude sector nearest the terminal area. For HITL simulations, surrounding airspace is managed by confederate 'ghost' controllers.

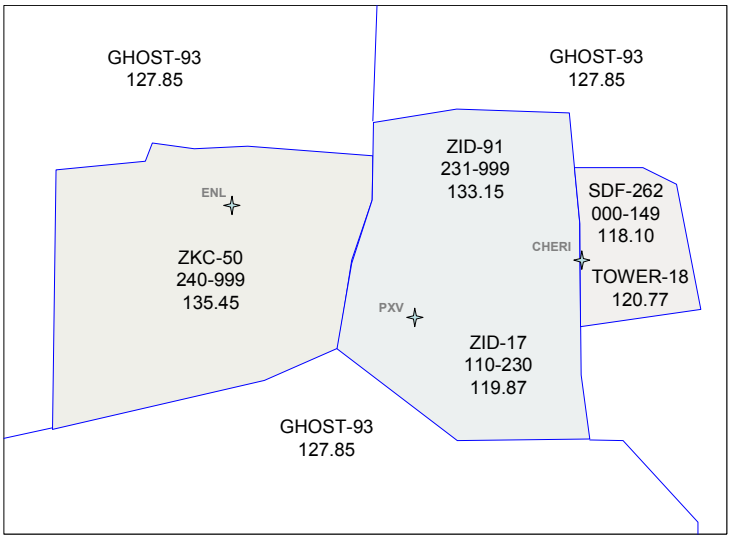

Figure 1. Test airspace for the example concept.

High-altitude sector controllers descend aircraft arriving from the west on the CDA routes shown in Figure 2. Test traffic scenarios have been developed that include arrival flows in which aircraft from participating airlines constitute the majority of arrivals. Controllers are responsible for fitting other arrivals into the arrival streams. Traffic scenarios also include varying levels of crossing traffic, some of which can interfere with CDA execution if not properly controlled.

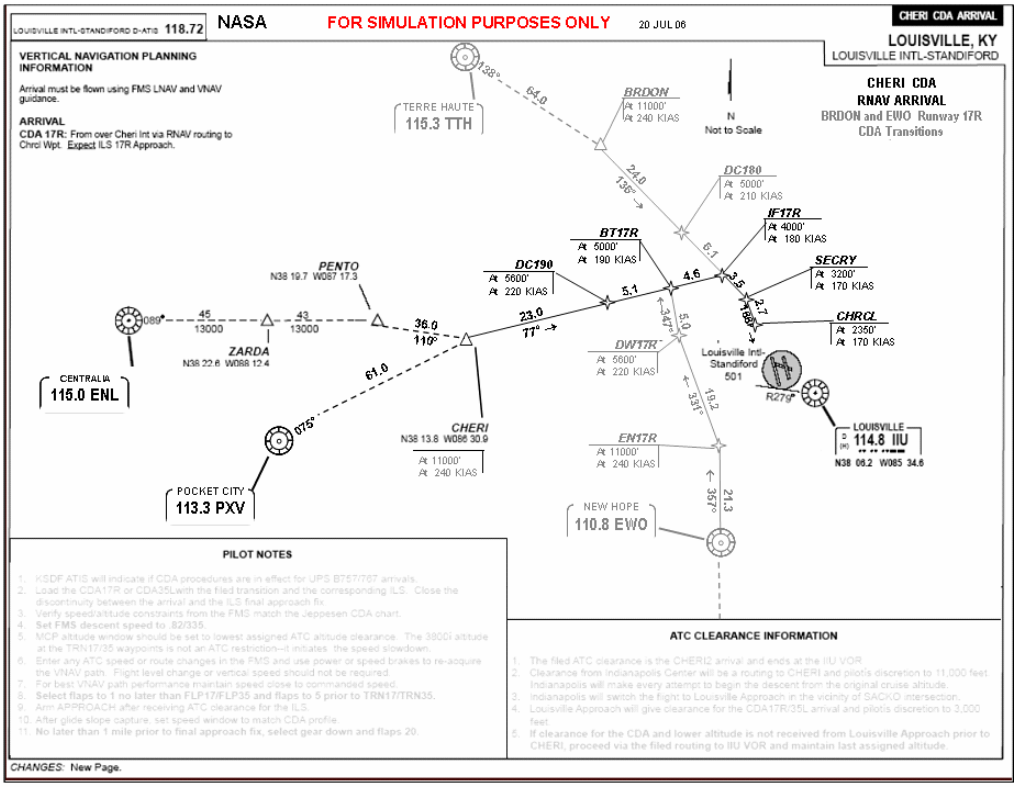

Figure 2. Example CDA arrival routes.

Air-ground HITL simulations are preliminarily designed to compare operations with three levels of ground-based automation tool support, in conditions with and without company aircraft from participating AOCs equipped for airborne separation assistance. The first level is current-day operations, in which controllers have no access to schedule timelines or other tools and participating AOCs condition arrival flows. The second level adds shared runway schedules and trajectory-based arrival metering tools; the third level adds the capability for controllers to data link trajectories to appropriately equipped aircraft. The HITL simulations seek to provide capacity data in terms of arrival throughput metrics, efficiency data in terms of potential CDA flight time realized, and safety data related to separation violations, including wake vortex spacing violations at the runway threshold—in addition to data to support iterative concept refinement from a human-systems integration perspective. Important data for this purpose include feedback from pilots and controllers about workload and coordination issues, operational acceptability, coping strategies, and the acceptability and use of prototype automation.

IV. Agent-Based Analysis

Research questions about air traffic controller operations for the example concept include questions about interference of crossing traffic with CDA arrivals, and how effectively en route controllers can manage arrivals from non-participating airlines in concert with AOC speed adjustments to company aircraft. For the agent-based analysis, AOC and flight crew roles are treated implicitly: participating AOCs issue appropriate cruise speed advisories to company aircraft and flight crews comply with clearances in nominal time. In the current TCSim instantiation of the concept, the AOC sequencing and spacing function has been automated, but the MACS airborne spacing algorithms are not currently implemented. Thus, the agent-based analysis will initially focus on scenarios in which aircraft are not equipped for airborne spacing in both near-term conditions and farther-term conditions with controller tools and data link.

In keeping with the analysis methodology, the agent-based analysis will focus on the sensitivity of different control strategies to environmental factors, and tradeoffs that arise, as measured by capacity, efficiency and safety metrics. The analysis will use the 'positive separation' of aircraft as a safety metric, in addition to separation violations. Positive separation refers to whether aircraft will maintain separation in the absence of further intervention on the part of controllers. In this concept, it is important for CDA arrivals in relation to crossing traffic.

The agent-based analysis will systematically vary two environmental factors: traffic characteristics and winds. Traffic characteristics include characteristics of arrival flows and characteristics of crossing traffic. The analysis will vary arrival flows along four interrelated dimensions: density of arrival flows, percentage of aircraft subject to AOC flow conditioning, amount of schedule delay to absorb, and percentage of terminal-area merges. As the number of arrivals en route controllers are responsible for increases, the more sequencing and spacing work they will have to do. In addition, in conditions without arrival timeline tools, the more difficult the sequencing and spacing task will be. The amount of delay individual aircraft will need to absorb also affects performance, as there is a limit on how much delay can be absorbed with cruise speed adjustments alone within the en route airspace. If controllers do not have time to assign speeds early enough, aircraft may require vectors or, in conditions with trajectory-based tools, lateral route amendments. Absorbing delay in descent can affect whether and how aircraft fly CDAs. The amount of work required to manage merging CDAs depends on suitable control methods and what controllers know about required merges. In near-term conditions terminal-area merges are likely to be problematic, because upstream controllers will not know where to leave slots in the arrival streams. In dense arrival flows, more terminal-area merges may result in more cancelled CDAs.

The analysis will also vary three interrelated characteristics of crossing traffic: overall traffic level, percentage of over-flights versus departures, and percentage of crossing traffic at altitudes that may impact CDA arrival profiles. Traffic crossing through airspace regions with arrivals on CDAs has greater safety implications and may be more difficult to control. Departures, especially aircraft departing to the west in the test airspace, may be on routes that share waypoints with arrivals. Some departures may interact with arrivals both laterally and vertically.

The second environmental variable is winds. The analysis will consider how differences in predicted and actual winds affect the runway schedule and operations that rely on it. Large differences have the potential to degrade the effectiveness of AOC flow conditioning, as well as arrival flow management by controllers using timeline tools. Winds also affect how readily aircraft respond to particular clearances. For example, depending on their direction and speed, winds can help or hinder aircraft deceleration.

The agent-based analysis will test the effects of environmental variations on the effectiveness of different control strategies. Strategies will range from simple (e.g., only adjust the speed of arriving aircraft) to complex (e.g., first try speed, then altitude, then lateral clearances). Preliminary studies have already identified sector-specific differences in required control strategies. For example, while the ZKC-50 controller (Figure 1) may prefer using speed clearances to adjust arrival spacing, the ZID-17 low altitude controller may need to use lateral clearances to control arrivals merging at CHERI (Figure 2). The SDF-262 controller also requires the option to reassign aircraft to runway 17L, should they end up poorly spaced for the terminal-area merge.

Control strategies also specify how interactions between different classes of aircraft (e.g. arrival vs. over-flight) affect the choice of control method and which aircraft to clear. Preliminary studies have indicated altitude clearances are a powerful tool for ensuring positive separation and avoiding the need to vector. Altitude clearances are also easier to implement than vectoring, because vectoring requires monitoring for when to issue a turn-back. The agent-based analysis will provide details on the traffic characteristics and winds under which particular control strategies are effective.

Left out of this description so far is the issue of individual aircraft types. Aircraft types are an important facet of arrival streams that affect scheduling, CDA profiles, and maneuverability (and therefore control strategies). The appearance of an ‘odd’ type, such as one that is not equipped to fly a CDA using flight management automation, can make controllers work harder, because it has to be handled differently. The effects of aircraft type mixes, like mixed aircraft equipage, is an important area for analysis.

The next section describes ATC agents implemented within TCSim. After detailing the agent model and architecture, the paper addresses the issue of integrating the agents in MACS.

V. ATC Agent Model and Architecture

The ATC agents are implemented in the Java programming language. The basic approach to integrating ATC agents with an ATM simulation is depicted in Figure 3. Each ATC agent has access to information on the traffic display and can issue clearances to aircraft. When tools are available, agents query them for information about sequence, schedule, conflicts, and other information that they would otherwise need to infer from the displayed traffic. Agents are updated at a rate that enables them to perform all the necessary processing without falling behind the simulation.

The agent model is actually a collection of models that enable it to perform ATC functions. Individual models include:

1) High-level executive model to control processing;

2) Sector-specific knowledge model;
3) Model of control strategies to apply;
4) Agenda formulation/task management model;
5) Models for implementing individual control methods;
6) Model of controller's 'picture' and the processes required to construct and maintain it;
7) Models of coordination with other agents, including aircraft ownership and clearance histories;
8) Activity timing models;
9) Models of automation tool usage.

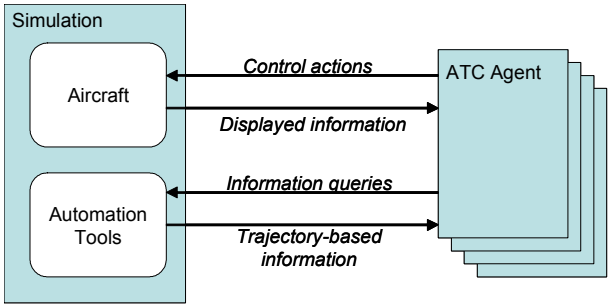

Figure 3. Agent-to-simulation information exchange.

The subsections that follow describe objects central to the ATC agent architecture. To support this discussion, an overview of the processing that occurs within an ATC agent is helpful. Figure 4 shows a hierarchical flow diagram of the ATC agent processing scheme. At left are the three activities represented in the high-level executive model. These activities are performed cyclically. The first activity, 'Assess Traffic,' includes the processing necessary to update the agent's picture of the current traffic situation. As has been hypothesized for human air traffic controllers, this activity requires the bulk of processing. The picture contains all the agent's predictions about where individual aircraft are going, as well as predictions about where they will go if the agent performs certain control actions (i.e., issues certain clearances). These predictions are referred to as planned trajectories ('Planned Trajs') in Figure 4. Plans for future activities in the picture include control plans, which represent the instances of control method(s) the agent is slated to try first, according to characteristics of the agent's control strategy and a particular problem.

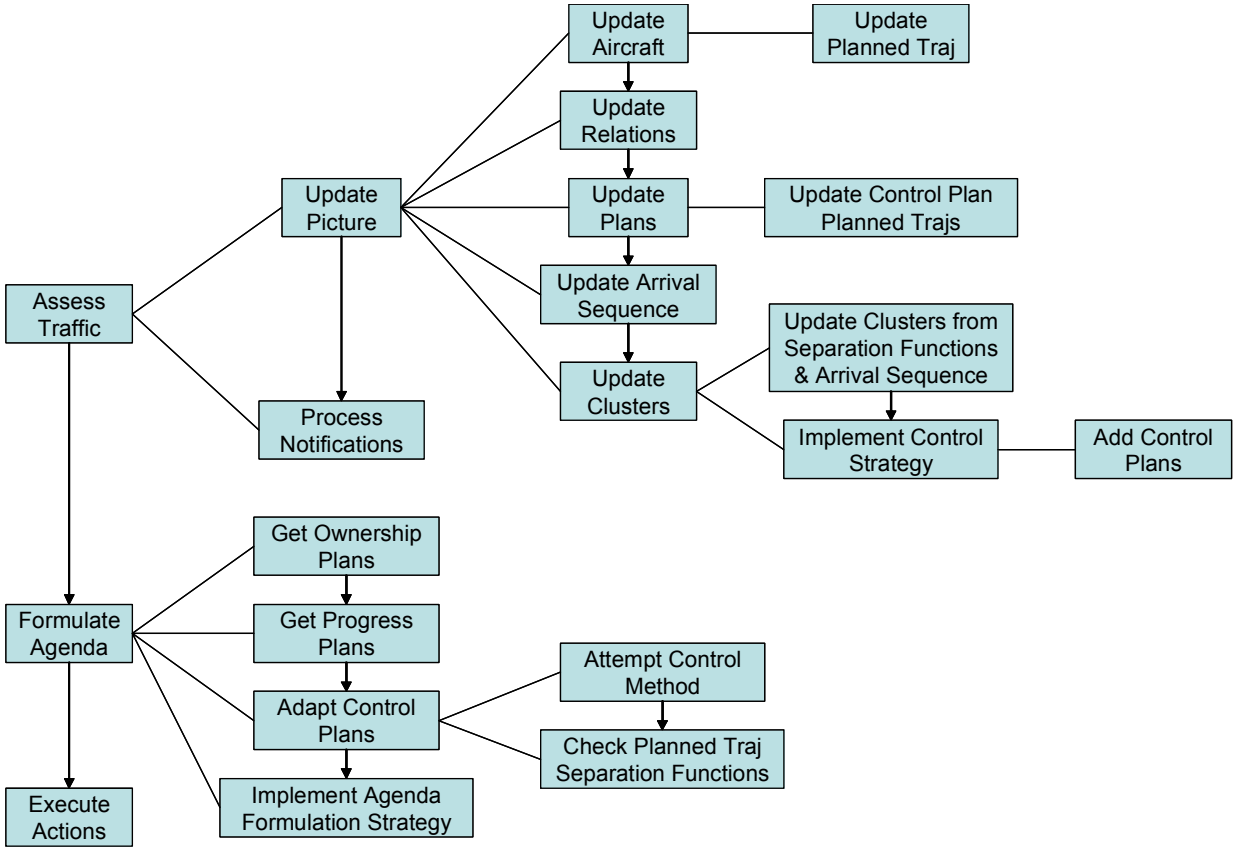

Figure 4. ATC Agent processing scheme.

A. ATCAgent

The top-level class in the agent architecture is ATCAgent (Figure 5). An instance of ATCAgent contains the agent's name, as well as an instance of SectorKnowledge (subsection B), an instance of Picture (subsection E), a high-level executive model, a master list of plans that the agent has executed, a list of notifications from other agents to process, as well as graphical display and data logger instances and timing data.

ATCAgent performs actions according to the state of a high-level model that controls processing. The invoke() method updates an agent's timing data and, based on the amount of time available for processing until the next update, performs required actions. The three methods assessTraffic(), formulateAgenda(), and executeAction() correspond to state transitions in highLevelModel. Each of these methods updates the remaining time available; overruns are subtracted from time available in the subsequent processing cycle(s). The assessTraffic() method updates the agent's picture, as shown in Figure 4. This activity takes an amount of time that depends on the number of aircraft visible on the agent's display. The formulateAgenda() method collects planned actions ready to be performed from the agent's picture, and applies agenda formulation strategies to the list of actions to arrive at a final agenda of actions to execute. This action is also assumed to have a duration that depends on the number of actions in the agenda. The agenda formulation strategies can be specified to ensure that the number of agenda items does not exceed some number (e.g., five actions).

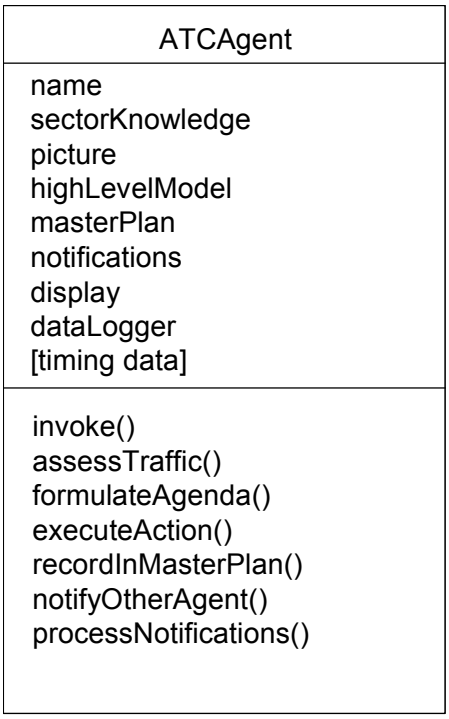

Figure 5. ATCAgent class.

The executeAction() method executes each action on the agenda in turn. As it does so, it executes the recordInMasterPlan() and notifyOtherAgent() methods. recordInMasterPlan() copies the executed plan from its parent plan (described in subsection C below) and records it in masterPlan, creating an accounting of what the agent has been doing at every point in time. Actions that require notifying other agents, such as handoff initiations or acceptances, are recorded on the other agent's notifications list. When the other agent updates its picture, it executes processNotifications(), which updates the agent's aircraft ownership model or clearance history for the aircraft in question. The notification scheme is used to model coordination between air traffic controllers that happens via the controller's display, such as 'flashing' a handoff to a downstream controller, as well as that which happens verbally (e.g. "AC123 is on a 160 heading"). The latter sort of coordination only applies in situations where controller positions are modeled to be located next to one another.

B. SectorKnowledge

A SectorKnowledge object (Figure 6) is instantiated for each agent by parsing a text configuration file. Member variables include sectorNumber, frequency, separation minima (including spacing target values, where applicable), and the agent's area of regard, which represents the airspace visible on the agent's display. SectorKnowledge also includes three knowledge representations: exitConditions, agendaFormulationStrategies, and controlStrategies.

exitConditions represents some of the information found in Letters of Agreement with neighboring sectors. For example, aircraft entering a sector N should do so by crossing the entry fix XYZ at 240 kts and 11,000 ft. agendaFormulationStrategies specifies how agents are to prioritize actions on the current agenda. controlStrategies specifies how agents should apply control, as described above.

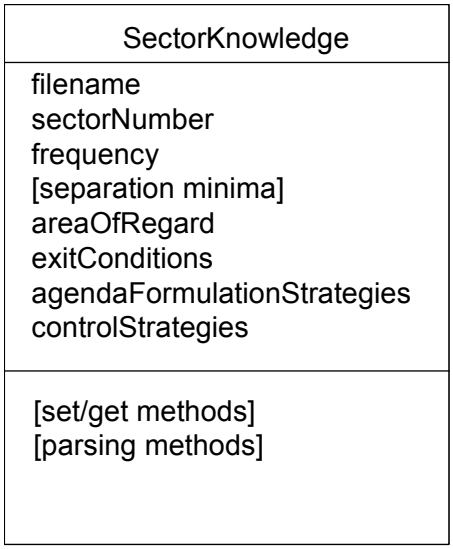

Figure 6. SectorKnowledge class.

C. ACInformation

ATCAgent represents knowledge about individual aircraft with instances of the ACInformation class (Figure 7). The first member

variable, aircraft, is a link to the aircraft's representation in the host simulation. Through this variable, an agent has access to information about the aircraft that appears on the controller's display. Using methods accessible through the simulated aircraft (e.g. schedule entries, trajectory-computation methods), an agent can simulate the acquisition of information that would be provided through automation tools. The atcAgent member variable enables methods within the ACInformation class to access sector knowledge and timing data contained in the ATCAgent class.

A critical variable is plannedTraj, a representation of the trajectory a controller is planning for an aircraft to fly. Controllers are modeled to mentally construct a planned trajectory for each aircraft from state information presented on the traffic display and flight plan information on the aircraft's flight strip. The planned trajectory may be simulated at different levels of fidelity that reflect assumptions about how controllers construct it. For example, aircraft on FMS descents with speed restrictions may perform decelerations that controllers approximate with average speeds along segments. When trajectory-based tools are available, however, controllers may display trajectories down-linked from aircraft to produce a more accurate picture of their trajectories. In these cases corresponding accurate representations of plannedTrajs can be computed using trajectory-computation methods in the host simulation.

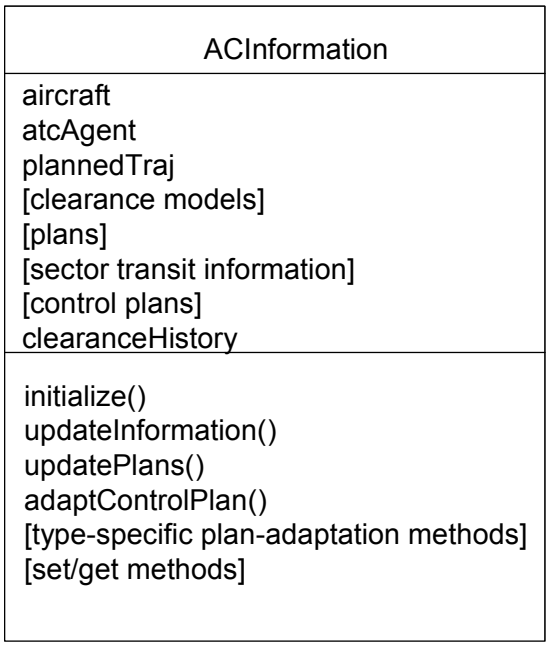

Figure 7. ACInformation class.

ACInformation also has clearance models that represent the states aircraft traverse as they perform maneuvers in response to particular clearances (Figure 8). For example, if a controller issues a heading vector, the aircraft transitions from the 'charted route' state to the 'heading vector' state. If the controller later issues a direct-to

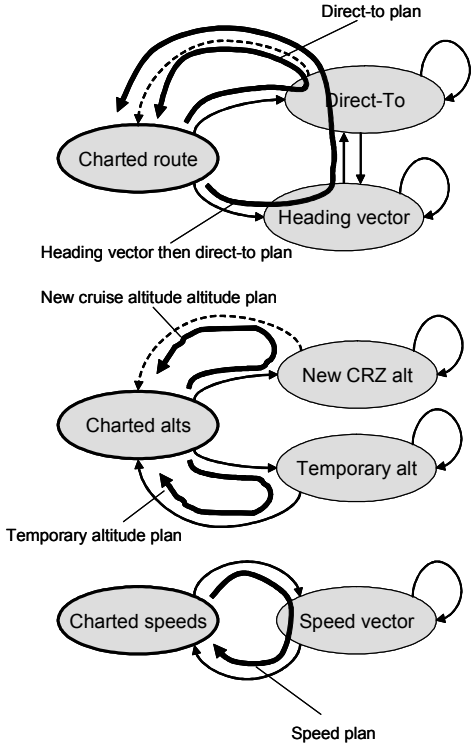

Figure 8. Example clearance models.

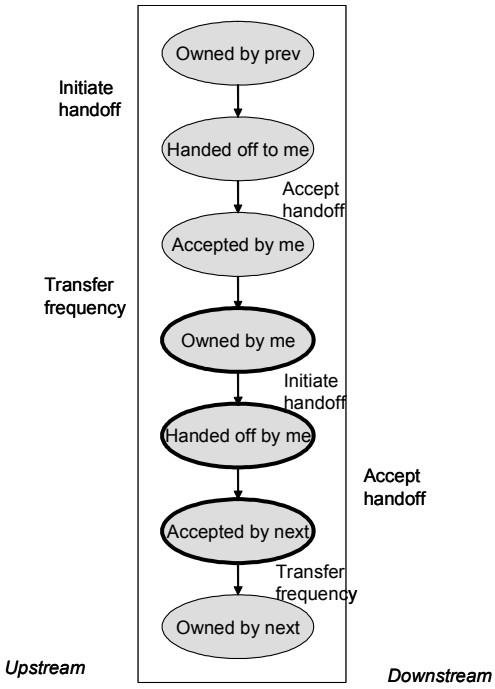

Figure 9. Ownership model.

clearance to a waypoint on the aircraft's route, the aircraft transitions to the 'direct-to' state. When the aircraft reaches that waypoint, the aircraft transitions back to the 'charted route' state.

ACInformation also includes a set of plans devoted to ownership and aircraft progress. Ownership plans represent the states and activities agents perform to acquire and transfer track control. They are derived from the ownership model shown in Figure 9. The ownership plans for a particular aircraft depend on the sector transit information contained in ACInformation, which is determined using its plannedTraj. The sector transit information determines which other agents represent the upstream and downstream agents. These are the agents with which the agent exchanges notifications upon performing ownership activities, so that those agents can update the status of their ownership plans for the same aircraft. If no agent is staffing an upstream or downstream sector, as may be the case for 'ghost' sectors, a simplified ownership model is used in which no explicit transfer of control is required—the 'accept handoff' and 'initiate handoff' actions enable the agent to acquire ownership, or transfer ownership, of the aircraft. Another special case involves aircraft that never leave the agent's airspace. When the aircraft is in the states outlined in bold in Figure 9, the aircraft is controllable by the agent. Radio check-in's from aircraft are missing from the ownership model in Figure 9, but could be easily added immediately following the 'transfer frequency' action from the upstream controller.

Aircraft progress plans are also derived using the plannedTraj. Progress plans tell an agent when to issue descent, approach transition, approach, or landing clearances to a given aircraft. In the test airspace shown in Figure 1, certain arrival aircraft may require a descent clearance from the ZKC-50 agent, while others can receive this clearance from the ZID-91 agent. The ZKC-50 agent therefore notifies the ZID-91 agent when it issues descent clearances, so the ZID-91 can focus on the aircraft that have not yet received one.

ACInformation also contains control plans that the agent has formulated for the aircraft. As shown in Figure 8, control plans are derived from the clearance models by tracing paths through them. Control plans are also distinguished as pertaining to lateral, vertical, or speed clearances. Depending on the control strategies an agent implements, aircraft can have more than one type of control plan pending at a given point in time. As shown in Figure 4, agents plan control actions dynamically rather than immediately reacting to potential separation or spacing problems; an agent adds control plans during its picture updating process, then attempts to implement them during the agenda formulation process.

Important methods in the ACInformation class include initialize(), which is responsible for constructing the aircraft's initial plannedTraj and establishing ownership and flight progress plans. updateInformation() updates the aircraft's plannedTraj on subsequent updates to an agent's picture. The updatePlans() method supports graphical display of control plans and updates the plannedTrajs that control plans contain to represent the planned flight path of the aircraft, given the maneuvers specified in the control plan. Like updatePlans(), adaptControlPlan() and sub-methods for adapting individual types of control plans are crucial for implementing a particular control strategy. Following the process flow diagram in Figure 4, an agent first applies its control strategies and attaches a control plan to an instance of ACInformation that represents an aircraft in conflict. During the agenda formulation process the agent collects all the activities that it could perform as part of the current agenda. It collects individual clearance plans by executing the adaptControlPlans() method for each instance of ACInformation in its picture. The adaptControlPlans() method determines which plan to adapt and applies a type-specific adaptation method to attempt to find a clearance of the prescribed type. If modifications to the aircraft's plannedTraj do not conflict with plannedTrajs of other aircraft, the process is successful and the clearance action is returned to the agenda formulation process for execution. Depending on the agenda formulation strategies in play, the agent may execute the clearance and establish a corresponding plannedTraj for the aircraft or perform the plan-adaptation process anew on the next agenda formulation cycle.

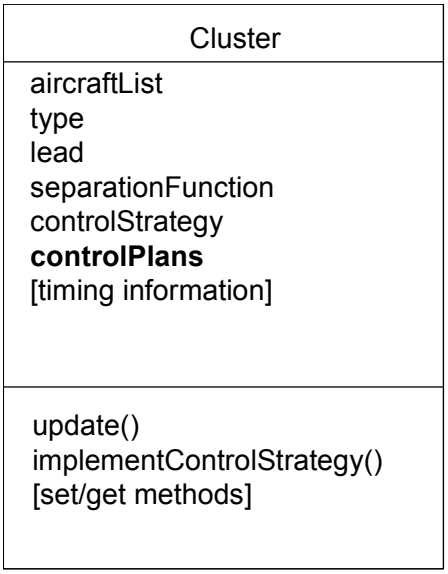

Figure 10. Cluster class.

D. Cluster

One additional class is important for understanding how an agent represents the controller's picture. The Cluster class represents a currently pending separation or spacing problem that an agent has identified (Figure 10). Cluster contains a list of the two aircraft involved, whether the cluster represents a spacing or separation problem, and the lead aircraft (if it is a spacing-type cluster).

separationFunction holds a so-called 'separation function' that represents the lateral and vertical separation between the aircraft's plannedTrajs up to some time in the future (e.g. 10 mins). An agent constructs a separation function for every pair of aircraft when it updates its picture; those that reveal potential problems are represented as clusters. Cluster updates occur at the end of a picture update, as shown in Figure 4. The update() method updates information in clusters that existed previously. Next, every cluster executes its implementControlStrategy() method, which assigns a control strategy from the set specified the agent's sectorKnowledge to the cluster and establishes the prescribed control plan for both the cluster and the ACInformation instance determined by the control strategy. The agent is now ready to attempt to adapt the control plan on the forthcoming agenda formulation cycle.

E. Picture

The model of the controller's 'picture' is represented by an instance of the Picture class (Figure 11). Previous subsections have already described key components of the picture. In addition to an atcAgent member variable that enables access to an agent's sectorKnowledge, the picture contains aircraftList, comprised of instances of ACInformation that represent the aircraft that are within the agent's area of regard. arrivalSequence represents an ordered list of ACInformation instances for arrival aircraft. When an agent has access to sequence or scheduling tools, the tools are accessed to create the list; when it does not, the agent constructs the list itself using information from the plannedTraj for each aircraft. clusters holds the current control problem clusters.

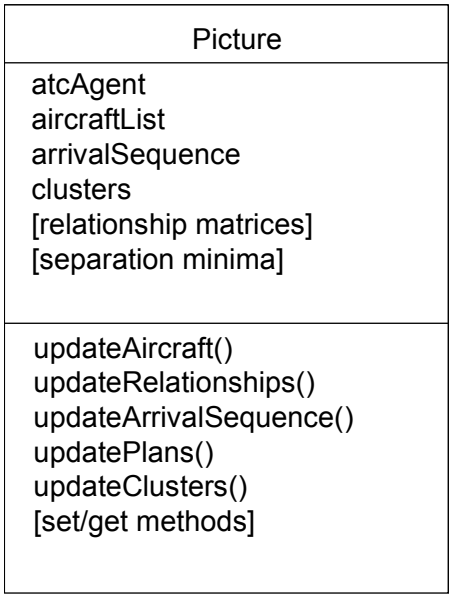

Figure 11. Picture class.

Separation functions are contained in one of three relationship matrices maintained in a Picture instance. A second relationship matrix holds information on the first waypoint shared by each pair of aircraft; a third holds information about whether each aircraft pair has the same downstream sector. The shared waypoint relationships are crucial for constructing an arrival schedule when no scheduling tools are available. The downstream sector information helps determine the risk of 'handing a deal' to a downstream controller (i.e. transferring control of aircraft that are at risk of losing separation in the downstream sector)—something that should not occur.

Picture contains methods executed through an agent's assessTraffic() method, as shown in Figure 4. updateAircraft() collects instances of ACInformation in the agent's area of regard and places them in the picture's aircraftList. updateRelationships() updates the relationship matrices containing the separation function, downstream sector, and first shared waypoint information. updateArrivalSequence() establishes the arrival sequence for the current update cycle. updatePlans() simply executes the updatePlans() method of each entry in aircraftList. Finally, the updateClusters() method uses the separation function relationship matrix to identify new clusters and update preexisting ones. It then executes the implementControlStrategy() method of each cluster to install suitable control plans based on an agent's control strategies. At the end of each update of an agent's picture, the agent is ready to attempt to implement control plans that have been added to clusters and aircraft contained therein. Throughout this process an agent's dataLogger dumps a trace of the process to an output file, so that context leading to the selection of a particular clearance may be analyzed.

F. Control Methods

A final point concerns the control methods agents apply when they adapt control plans. Because these are computational agents, deciding on a value for a particular clearance takes on a flavor similar to highly automated NGATS concepts, such as the Advanced Airspace Concept.[1] That is, a generate-and-test approach must be applied to check that planned clearances will have the desired effect. An important difference is that planned trajectories are purposefully simulated at a level of fidelity that reflects assumptions about how controllers construct them. The level of fidelity of the planned trajectories dictates how well the agents detect problems and choose clearances to solve them.

An agent can generate a variety of planned trajectories of varying complexity. For example, it can produce a planned trajectory that reflects vectoring an aircraft to some heading for a planned time or distance before turning it back to its charted route, or one that reflects a changing an aircraft's cruise altitude and issuing a subsequent

clearance to change back to its original cruise altitude after some time or distance. The generate-and-test process is represented by the two processes, 'Attempt Control Method' and 'Check Planned Traj Separation Functions' in Figure 4. The 'check' process involves comparing the planned trajectory generated for a particular clearance plan to the plannedTrajs of all other aircraft and ensuring there are no predicted losses of separation. The process is complicated when other aircraft are also in the process of executing multi-step maneuvers, because the plannedTraj that was used to check *their* plannedTrajs must be updated and used to check the clearance for the current aircraft.

For efficiency, efforts have been made to streamline the iterative process used to find workable clearance plans. For example, adapting a single-step vertical control plan for an aircraft in cruise flight first involves checking a planned trajectory for the next higher legal flight level, then the next lower legal flight level, then two flight levels above, then two flight levels below, etc. In general, all of these methods attempt to either shorten the aircraft's overall trajectory, or minimize deviations from it.

VI. Multi Aircraft Control System (MACS) Integration

Affording computational agents access to the same prototype automation tools and aircraft models used in HITL studies is the main motivation for integrating ATC agents in MACS. The successful integration of another intelligent agent, the Crew Activity Tracking System (CATS),[12] with MACS provides a useful foundation for the ATC agent integration work. Because MACS is also Java-based, different code bases are not an issue. This section discusses several MACS integration issues that deserve consideration.

First, MACS is a multi-threaded application with more than a hundred different threads. To achieve proper performance running within MACS threads, the ATC agents will need, at minimum, to use smaller processing units. More likely, they will require implementation as multi-threaded processes themselves. This raises a number of modeling issues, including the possibility of working from a partially updated picture of the current situation. A related problem concerns the traceability of multi-threaded agent behavior. Agent performance in the current TCSim implementation, while somewhat complex, is perfectly repeatable. Repeatability in a multi-threaded agent model may be difficult to achieve. Processing in CATS is somewhat simpler, but successful implementation of CATS as a multi-threaded process suggests it can also be accomplished for the ATC agents.

The database of information MACS uses for configuration will also need to be modified to include ATC agent configuration information. In the CATS implementation, only the CATS model of nominally preferred pilot activities was added. ATC agents will require sector knowledge and control strategy configuration files. ATC agents can use existing database information used to configure MACS controller workstations. This would enable the analyst, for example, to dynamically change the agent's area of regard by simply panning or zooming the display. All the controller tool settings are also included in the MACS database, so adjusting these settings can determine the tools configurations to be used by ATC agents.

Because MACS runs either as a single standalone process or as a distributed simulation, however, single-controller MACS station configuration information cannot be used exclusively. Threading schemes must also enable multiple ATC agents to run within a single MACS process. ATC agents currently interact only with other ATC agents that update synchronously. A related issue concerns partial human staffing for part-task studies or other simulations that are not fully autonomous closed-loop simulations. In MACS any coordination that occurs via the display does not require explicit notification. The notification process can still be used for direct agent-agent communications, but alternative schemes are required to enable ATC agents to communicate with humans. This problem also concerns interactions with human pseudo-pilots with whom controllers typically have the option of communicating with by voice. Similar issues were addressed during the MACS-CATS integration process and solved using MACS data link functionality. Autonomous MACS pilot stations use a similar scheme.

Accessing MACS automation tool functionality is straightforward. Because every instance of ACInformation that ATC agents initialize contains a reference to the host simulation aircraft, the ATC agents will have ready access to schedule information, trajectories, and the methods that support related computations. Existing MACS tool prototypes rely heavily on MACS trajectory computation methods and ATC agents can implement similar methods. For example, to model controller use of trial-planning tools, ATC agents can specify a list of new lateral trajectory change points, with other aircraft trajectory information (e.g. speed schedules, crossing restrictions) remaining the same, and MACS will compute a trajectory. Furthermore, MACS checks each trajectory to ensure that it is flyable and records this information in the object that represents it.

A key advantage is the more sophisticated flight guidance modeling in MACS. Aircraft performance is more robust to interventions, and guidance-based tools like airborne separation assistance are currently implemented. Therefore controller spacing advisory information is also readily accessible.

Displays, such as displays for monitoring agent processing, are easy to implement in MACS. MACS also has a facility for logging various data that includes configuration windows for selecting which data to collect. These capabilities are readily extensible to accommodate ATC agents.

VII. Conclusions and Further Research

This paper has described an ATC agent model and architecture and illustrated how it can be applied to analysis of NGATS concepts. The model is a task-analytic worksystem model that supports an analysis methodology designed to complement HITL simulations. The analysis methodology focuses on examining control strategy effectiveness to be tested in the presence of different technologies and environmental factors. Results support the evaluation and refinement of human roles and responsibilities in future ATM systems. An important aspect of complementary fast-time and HITL simulations is consistency in aircraft and automation performance. To that end, ATC agents will be implemented in MACS.

Among areas that require further research is, first, the analysis of the example concept. Research is underway to develop traffic scenarios and a test set of sector-specific control strategies. These are efforts are due to be completed around the same time as HITL simulations so the results can be considered in parallel.

The second area is MACS integration—in particular, a multi-threaded ATC agent implementation and the capability to use ATC agents in various simulation configurations. An initial implementation will leverage as much of the currently implemented ATC agent architecture as possible. Work in this area has also begun, with some agent processing and display threads added to a research prototype of MACS.

Finally the specification of control strategies and control methods in the ATC agents represents an area of continuing research. Different assumptions about how controllers construct planned trajectories require implementation and testing. Moreover, an improved understanding of how these ATC agents relate to highly automated NGATS concepts and cognitive human performance models that represent air traffic controllers is warranted, considering the importance of agent-based concept validation studies for the NGATS.

Acknowledgments

Funding support for this research is provided by the NASA Airspace Systems Program.

References

[1]Andrews, J., Erzberger, H., and Welch, J., "Safety Analysis for Advanced Separation Concepts," *Air Traffic Control Quarterly,* Vol. 14, No. 1, 2006, pp. 5-24.

[2]NASA, *Research Opportunities in Aeronautics—2006*, NASA Research Announcement NNH06ZNH001, National Aeronautics and Space Administration, Washington, D.C., 2006.

[3]Callantine, T., and Palmer, E., "Fast-time Simulation Studies of Terminal-Area Spacing and Merging Concepts," *Proceedings of the 22nd Digital Avionics Systems Conference*, IEEE, 2003, 03CH37449 [CD-ROM].

[4]Prevot, T., Smith, N., Palmer, E., Mercer, J., Lee, P., Homola, J., and Callantine, T., "The Airspace Operations Laboratory (AOL) at NASA Ames Research Center," AIAA-2006-6112, American Institute of Aeronautics and Astronautics, Reston, VA, 2006.

[5]Prevot, T., Battiste, V., Callantine, T., Kopardekar, P., Lee, P., Mercer, J., Palmer, E. and Smith, N., "Integrated Air/Ground System: Trajectory-Oriented Air Traffic Operations, Data Link Communication, and Airborne Separation Assistance," *Air Traffic Control Quarterly,* Vol. 13, No. 2, 2005, pp. 201-229.

[6]Callantine, T., Lee, P., Mercer, J., Prevôt, T., and Palmer, E., "Simulation of Terminal-Area Flight Management System Arrivals with Airborne Spacing," *Air Traffic Control Quarterly,* Vol. 14, No. 1, 2006, pp. 47-67.

[7]Callantine, T., "Modeling and Simulation of Trajectory-Oriented Air Traffic Management Concepts," AIAA-2004-5260, American Institute of Aeronautics and Astronautics, Reston, VA, 2004.

[8]Callantine, T., Prevot, T., Battiste, V. and Johnson, W., "Agent-based Support for Distributed Air/Ground Traffic Management Simulation Research," AIAA-2003-5371, American Institute of Aeronautics and Astronautics, Reston VA, 2003.

[9]Dowell, J., "Formulating the Cognitive Design Problem of Air Traffic Management," *International Journal of Human-Computer Studies*, Vol. 49, 1998, pp. 743-766.

[10]Corker, K., and Verma, S., "Analysis of Advanced Airspace Concept Operations Using Human Performance Modeling," *Proceedings of the International Symposium on Aviation Psychology*, Oklahoma City, OK, 2005, pp. 602-607.

[11]Prevôt, T., Callantine, T., Kopardekar, P., Smith, N., Palmer, E., and Battiste, V., "Trajectory-Oriented Operations with Limited Delegation: An Evolutionary Path to NAS Modernization," AIAA-2004-6449, American Institute of Aeronautics and Astronautics, Reston, VA, 2004.

[12]Callantine, T., "Detecting and Simulating Pilot Errors for Safety Enhancement," SAE Technical Paper 2003-01-2998, Society of Automotive Engineers, Warrendale, PA, 2003.