*Appeared in IEEE Computational Intelligence for Security and Defense Applications, April 2007*

# Toward Automating Airspace Management

Glenn Taylor, Brian Stensrud, Susan Eitelman, Cory Dunham, Echo Harger
Soar Technology, Inc.
3600 Green Court, Suite 600
Ann Arbor, MI 481051

***Abstract*—Military airspace is increasingly crowded with traditional aircraft competing with new loitering munitions and UAVs. Managing the airspace is therefore more challenging, requiring closer coordination among all the stakeholders. In this paper, we describe the motivation and design of a knowledge-based system that attempts to automate aspects of airspace management, including the detection and resolution of airspace conflicts. We then describe a formative evaluation of the system as compared to human performance of the same task, the evaluation setup, results, and analysis.**

***Index Terms*—Airspace Management, Air Traffic Services, Intelligent Agents, Knowledge-Based Systems**

## I. Introduction

THE US military and other airspace control authorities are facing a growing problem of overcrowded airspace as more users make increasing demands on a limited resource. In the modern battlespace, traditional aircraft have always competed with traditional artillery for airspace, but now they also compete with modern weapons such as persistent air munitions (PAMs), loitering air munitions (LAMs), and a wide variety of UAVs, many of which are not under the direct control of the airspace authority. One function of Army Aviation is Airspace Management, which has to do with "the coordination, integration, and regulation of the use of airspace of defined dimensions" [1]. The Army continues to look at simulation-based exploration of Army Airspace Command and Control (A2C2) processes and policies in order to more efficiently use the airspace and improve mission execution overall.

This paper describes an effort to automate the Air Traffic Services (ATS) component of A2C2, which deals, in part, with ensuring that aircraft work within their assigned spaces and do not conflict with each other. The general hypothesis guiding this work is that we can automate the ATS elements of Airspace Management to a level equal to or surpassing human performance, reducing the costs of employing ATS within simulation environments.

This work was supported in part by the U.S. Army under Grant W911W6-05-C-0035. Gratitude to Mr. Jeff Maddox, Mr. Tim McKelvy, and others at AMRDEC, for support on this program.

Glenn Taylor is with Soar Technology, Inc., Ann Arbor, MI 48103 (phone: 734-327-8000; fax 734-913-8537; glenn@soartech.com); Cory Dunham a Software Engineer with Soar Technology's Ann Arbor office (duhnam@soartech.com); Echo Harger is a Project Manager with Soar Technology's Ann Arbor office. (echo.harger@soartech.com)

Brian Stensrud, PhD, and Susan Eitelman are with Soar Technology's Orlando, FL office. (stensrud@soartech.com, susan.eitelman@soartech.com)

## II. Description of the Problem Space

The Air Traffic Services (ATS) element of Airspace Management includes keeping track of where aircraft are at any given time (flight following), maintaining situational awareness (SA) of the airspace, providing SA to aircraft in the airspace, and detecting and resolving conflicts. (ATS is distinct from more typical Air Traffic Control, which is often more narrowly focused on terminal operations around airports. We will use the term 'controller' to refer to any person, in ATS or ATC, who is giving direction to an aircraft while it is flying.)

For both ATS and ATC, the controller's job is a knowledge-intensive task requiring constant updates to the "mental picture" the controller maintains in order to properly manage the airspace. The controller knows which aircraft are currently, or will soon be, flying through his or her assigned airspace, the particulars of the mission each aircraft is flying, all the routes and corridors (called Airspace Control Measures, or ACMs) of interest within the airspace, the current air picture, the standard operating procedures of the airspace (such as minimum separation rules), and basic doctrine of airspace management, command and control, and airborne operations (aircraft types and restrictions).

Our focus in this paper is scoped to the detection and resolution of airspace conflicts. There are two basic types of conflict of interest:

**Air-to-Air conflicts** – when two aircraft do not maintain the minimum separation rules, or seem to be on trajectories that will violate minimum separation rules in the near future.

**Air-to-ACM conflicts** – when one aircraft goes outside its assigned Airspace Control Measure (ACM), or wanders into an ACM for which it is not cleared.

For our purposes here, ACMs are subdivisions of the physical space that are used to coordinate movement within the airspace. These include boundaries, corridors, routes, control points, areas or zones, etc. The Airspace Control Order (ACO) defines the approved ACMs for a period of time, which are in turn used in the Air Tasking Order (ATO) for each individual aircraft to define which routes or control points will

be included in a particular mission.

Upon detection of an Air-to-Air or Air-to-ACM conflict, the ATS element issues an advisory to the relevant aircraft, and the aircraft (presumably) responds both verbally and by adjusting its flight profile. For rotary-wing aircraft (RWA) operations (i.e., helicopters), much of the work of situational awareness is up to the pilots, in a kind of "see and be seen" mode of operation. Conflict resolution often takes the form of simply notifying the helicopter pilot of where nearby aircraft are flying, providing enough awareness to the pilot to take corrective actions within the bounds of his flight constraints. With human controllers and human pilots, an advisory would take the form of a radio message. An example might be *"Foxtrot95, this is Adam15, you are left of course … correct heading is 090."*

## III. Approach

To automate conflict detection and resolution tasks, we have taken a knowledge-based systems approach that integrates the various kinds of knowledge required to perform these tasks. Specifically, we have adopted the *knowledge-intensive agents* (or "heavy agents") approach, such as described in [2], in which our agent has goals and beliefs about its environment, and commits resources in terms of attention and appropriate knowledge to achieve its goals. A second guiding principle of the design was inspired by Endsley's notions of Situation Awareness (SA) [3], which guided the content, organization and processing of knowledge. Both of these are described in more detail below.

### A. Situation Awareness

Endsley's theory of Situation Awareness [3] is a guiding principle for the design of this system. Endsley [4] defines SA as the perception of the elements in the environment within a volume of time and space, the compression of their meaning, and the projection of their status in the near future." SA (i.e., a set of knowledge) is the product of a process called Situation Assessment, which is the information processing necessary for decision-making.

Endsley proposes three levels of Situation Awareness. The first level of SA involves the current situation: "…the perception of the elements in the environment within a volume of time and space." [1] For our system, this level of awareness is maintained by processing and storing up-to-date spatial information about relevant object in the battlespace – in this case, aircraft and ACMs. The second level – "the compression of their meaning" – relates current perceptual awareness to one's goals. Working toward the primary goal of avoiding conflicts, the system must determine whether conflicts exist between aircraft and ACMs by inferring relationships among these objects as they exist in space. For example, the system must be able to compute the spatial distance between two aircraft in order to assert if they are complying with or deviating from separation rules. The third level of SA pertains to projecting the status of the aircraft into the immediate future, and making determinations of whether conflicts will exist within that projection. For instance, if two aircraft are a safe distance apart but are on a collision course, it is important that the system be able to identify that future conflict and work to correct the situation.

From the perspective of developing a model of human behavior, the three levels of Situation Awareness defined by Endsley provide us a clear methodology by which to organize the knowledge needed to make these decisions, as well as the basic processes of *perception*, *comprehension*, and *projection*. In the case of this system, the decisions include the determination of when aircraft are in conflict with each other and with specific ACMs in the battlespace.

### B. Knowledge-Based Systems

Since decision-making in this arena involves perceiving and analyzing several types and sources of knowledge, it was natural to construct the system as a knowledge-based system that brings to bear knowledge to solve problems. A knowledge-intensive agent [2], which embodies a knowledge-based system into an agent framework, exists in a dynamic environment, has desires, goals, justified beliefs and assumptions, and can make plans to achieve goals based on its current beliefs about the environment. Additionally, a knowledge-intensive agent constantly manages its beliefs and goals while executing plans to achieve its goals.

From earlier analyses of Air Traffic Control [5, 6], a controller uses a great deal of static and dynamic knowledge to maintain situational awareness and make decisions. Static knowledge includes background knowledge about particular aircraft (e.g., flight dynamics and weight class), local terrain, military airspace doctrine, and standard operating procedures for a local area. Dynamic knowledge includes the current and projected air picture, a particular aircraft's mission, environmental conditions, and any outstanding requests by aircraft or by the controller. The controller must constantly update his or her "mental picture" with current knowledge to control the airspace. Given the knowledge requirements of ATS and the processes for maintaining Situational Awareness the knowledge-intensive agent approach is very fitting.

## IV. AutoATS System

In earlier work [7], we developed a prototype automated air traffic controller that could interact with pilots via a voice interface to perform flight following and minimal terminal operations. AutoATS builds on that implementation to incorporate capabilities for conflict detection and resolution. The AutoATS architecture consists of four major pieces:

### A. A knowledge-based decision-making component

The knowledge-based decision making component is embodied in a Soar agent that combines knowledge about aircraft missions, routes, current and future situations to detect and assess conflicts and issue advisories. Soar is a cognitive architecture designed for building knowledge-intensive agents [8]. Soar provides a uniform knowledge representation scheme for long-term knowledge in the form of forward-chaining production rules. Soar agents also possess a graph-based

working memory that contains symbolic representations of the agent's current beliefs and assumptions about its situation. Production rules operate on working memory to generate new assertions or retract old ones as part of managing beliefs, goals, and plans. In terms of the three levels of SA, the AutoATS agent uses working memory to maintain first level information about airspace elements (ACMs and aircraft) as well as assertions at the second and third levels. The logic required to generate the assertions, including the identification of conflicts, lies within productions.

### *B. Spatial Computer*

While the Soar agent stores and reasons over qualitative symbolic-level information about the airspace, quantitative calculations about elements in the airspace are offloaded to a Spatial Computer. As elements appear and move around in the battlespace, the Soar agent publishes and updates their position and heading information to the spatial computer. When the agent must determine a relationship between elements, it does so by making the necessary query. For instance, to determine whether two aircraft are in conflict, the agent must first determine their separation. That computation is made by the spatial computer then returned to the agent for analysis. The return to the agent may be in a qualitative form (e.g., aircraft Eagle13 is *within* Corridor34). The actual location in three-dimensional space is typically not important to the AutoATS agent, only whether the aircraft is inside or outside the ACM of interest.

### *C. A Situation Awareness Console*

The Situation Awareness Console (shown in Fig. 1) displays both the state of the battlespace (e.g., current aircraft position, details about their orientation and movement, current ACMs, etc.) and a list of conflicts identified by the AutoATS system. The console allows us one way to compare system performance to human performance data visually, in addition to log-based comparisons. In the future, this console will also provide deeper insight into the deliberation of the AutoATS agent in order to explain its actions in context.

### *D. Network Interface Layer*

One integral component of the AutoATS system is a network interface layer that reads in simulation data and uses it to populate the agent's beliefs about the airspace. Using Mak's VR-Link product, AutoATS is capable of operating using both HLA or DIS simulation protocols. VR-Link works out of the box with basic HLA RPR-FOM and DIS5.0, but, as with this exercise, may need to be tailored to alternate protocols.

Generally speaking, the AutoATS system operates by 1) receiving input about simulated aircraft, 2) updating the agent's Situational Awareness in each of the three levels, 3) determining if there are any conflicts based on the situation, and 4) issuing advisories where needed.


Fig. 1. Situation Awareness Console

Of particular interest to our work here is the corridor and the Airspace Control Point. An ACM corridor is defined as a sequence of connected 3D boxes in space that have width extents (distance from a centerline) and height extends (minimum and maximum altitude in the airspace). Deviation outside this corridor may constitute a kind of conflict. An Airspace Control Point (ACP) is simply a point on the ground (usually referring to an easily identified landmark in the terrain). Corridors and waypoints were the primary means of coordination of flights within the experiment described later.

Aircraft are thought of as existing within a minimum separation bubble. The bubbles are defined in the AutoATS system as oblong volumes extending forward in space to an extent based on the speed and direction of travel of the aircraft. If the bubbles of two aircraft overlap, those aircraft are in conflict. Air-to-Air conflicts may not exist in the current situation, but may potentially exist in the near future. The AutoATS agent estimates where the aircraft will be in the near future (5 and 10 seconds forward) based on their velocity and bearing, then computes whether their projected positions (their surrounding bubbles) overlap to cause a conflict.

The notion of conflict is not based solely on geometry – for example, an aircraft going outside the bounds of a corridor is not by itself a conflict. The aircraft may be leaving the ACM to transition to another phase in the mission (e.g., to land) or in response to a directive from the controller to avert another conflict (e.g., change altitude to avoid other aircraft of higher priority). Extra mission information is needed to determine whether or not an advisory needs to be generated. Kinds of extra mission information might include:

- Dialogue between pilot and controller that has established new goals or new expectations about the aircraft's mission
- Recognized intent based on knowledge about the aircraft's movement toward or away from ACMs.
- Mission changes, including changes to assigned routes or generation of new ACMs

Prior SA studies of Air Traffic Control [5, 6] indicate the goals of the controller, and relate those goals to the knowledge required to achieve them. In the AutoATS system, there are three broad types of goals: those that help maintain SA, those that respond to conditions in the battlespace by identifying conflicts, and those that manage generating and issuing advisories. By maintaining the three levels of SA, the agent is able to detect when conditions exist for a potential conflict. When this is the case, the agent generates a new goal for examining that condition and determining whether it is a conflict. If it is, the agent constructs an advisory that is displayed on the Situation Awareness Console. Advisories are generated using a simple template-based approach to natural language generation, which for now capture only the basic information of who-what-when-where.

## V. EVALUATION

A recent exercise conducted by the Army Aviation and Missile Research, Development, and Engineering Center (AMRDEC), called Joint Aviation Missile and Unmanned Systems 2006 (JAMUS2006), provided an opportunity for us to evaluate AutoATS against human data in a simulated environment. While the core experiment was not focused on interaction with aircraft, the experiment included a set of excursions that allowed us to use the same experiment setup to gather data regarding aircraft conflicts and controller-aircraft interaction.

AutoATS is one component within the larger JAMUS2006 simulation federation. Other components include the MATREX/HLA server, the IDEEAS constructive simulation system, data collectors, and an Airspace Management Console (ASM Console) operated by a human ATS controller. Each component is an HLA federate, running within the MATREX HLA federation. The JAMUS2006 architecture is illustrated in Fig. 2. For our evaluation, the AutoATS system is swapped for ASM and the human ATS controller. The experiment is somewhat ahead of current fielded technology in a few areas, not least of which is the information available to the controller regarding aircraft awareness in the battlespace. In current Army operations, a controller would only rarely have continuous real-time updates about aircraft position; in JAMUS2006, the controller receives constant situation updates that indicate the position and orientation of aircraft.

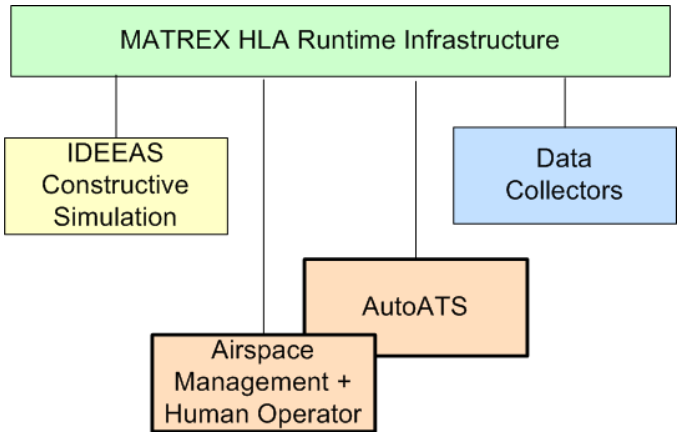


Fig. 2. JAMUS2006 System Architecture

### A. *Experiment design*

In this formative study, we separated the experiment into two cases: 1) human playing role of ATS; 2) AutoATS system playing role of ATS. In both cases, the ATS element's role was to detect conflicts and generate advisories to the relevant aircraft. We did not include artillery or other air munitions in these exercises (for example, no advisories were given relative to artillery passing through defined air corridors), but advisories would be generated for conflicts between aircraft and ACMs, and between multiple aircraft. In the first case, the human controller used the ASM Console that displayed aircraft position and orientation, airspace control measures (ACMs and ACPs) and, when an aircraft left an assigned corridor (the ASM did not detect air-to-air conflicts). The human controller used information available through the ASM console to generate advisories (such as which aircraft and which ACM), which were then spoken aloud and recorded. The ASM display was also recorded on video, to give later context in coding an analysis as to why an advisory was given. The simulation network data describing the flights of aircraft was also recorded.

In the second case, we used the recorded simulation network data to re-generate the situation, this time using the AutoATS system to detect conflicts and generate advisories. Because the aircraft behavior was entirely scripted, and the aircraft do not actually respond to advisories given by the human ATS player or AutoATS, the recording is no different than the live run exercise, and allowed us to use the recorded network traffic to run against the AutoATS system, and to perform other post-run analysis. The experiment setup is illustrated in Fig. 3.

We distinguish two important types of knowledge in each setup illustrated in Fig. 3: spatial knowledge (aircraft positions, separation bubbles, ACM locations and extents) and mission/situation knowledge (e.g., understanding of aircraft intent, prior interactions with aircraft, etc.). In the Human ATS setup, this knowledge is split across the human and the ASM console. The ASM handles spatial knowledge (and detects conflicts), and the human controller uses other knowledge to determine when an advisory should be given based on a conflict detected by ASM. In the case of AutoATS, both of these kinds of knowledge and processes are included in the single AutoATS system.

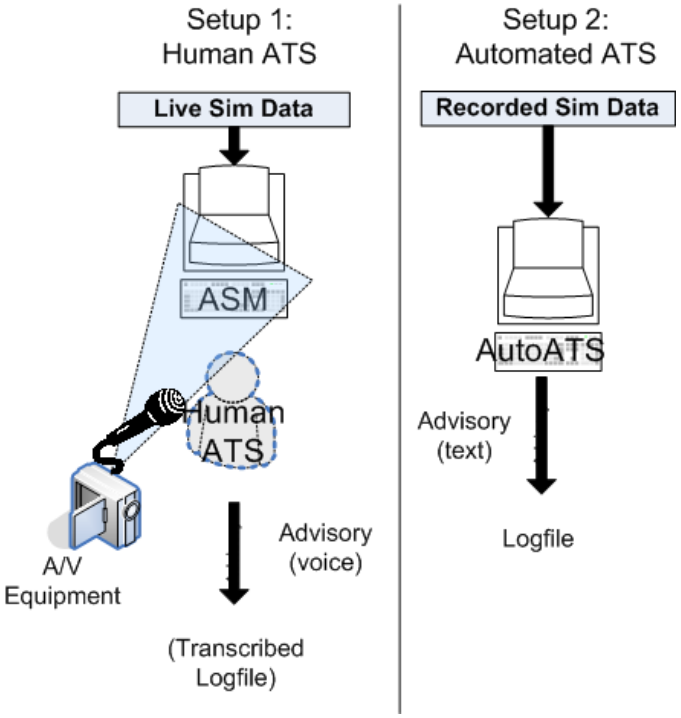


Fig. 3. AutoATS Evaluation Setup

There were two scenarios lasting 45 and 50 minutes respectively, with three primary missions in each scenario: a troop insertion by a flight of UH-60s, a mobile strike by a flight of AH-60s, and a MEDEVAC flight consisting of a single UH-60. There was also secondary UAV traffic in certain areas, and in one scenario, a pair of fixed-wing F16s fly through the airspace. In each of the scenarios, the aircraft were scripted to follow a set route, with varying amounts of maneuvering within and outside the assigned mission ACMs that would typically cause advisories to be generated. The maneuvering was inserted specifically to generate advisories – in the basic execution of the mission for JAMUS2006, the aircraft flew perfectly within the corridors, generating no advisories. For our purposes, we were less interested in perfect flights than in creating the conditions under which advisories might be generated.

As mentioned earlier, in no case were the aircraft responsive to controller communication, so the (lack of) effects of advisories are unrealistic. In real operations, if an aircraft flies outside a corridor unexpectedly, an advisory is typically generated, and the pilot would respond at least verbally, if not also by changing the flight path. However, the aircraft in the experiment do not change behavior in response to advisories, but instead fly according to the scripted flight plan. As such, the aircraft might in fact behave in a way that would normally cause another advisory to be generated. While perhaps unrealistic, considering the goals of the experiment, repeatability was a core feature of JAMUS2006. Furthermore, this condition was constant in both the human and AutoATS controller iterations, so the input data are the same. Because of this simulation artifact, we classify interactions according to 'first infraction' and 'follow-up' to note that some may be artifacts of the aircraft not responding to the controllers.

### *B. Experiment Results and Analysis*

Tables 2 and 3 below summarize the ASM/human versus AutoATS results for detecting conflicts and generating advisories in Scenario 1. "Detected Events" refers to potential conflicts that were detected by the respective systems. For some of these, advisories were generated.

TABLE 1
SCENARIO 1 RESULTS: HUMAN VS. AUTOATS

| | **ASM/ Human Controller** | **AutoATS** |
|---|---|---|
| Detected Events | 6 | 5 |
| Advisories Generated | 5 | 5 |
| Advisory Difference | 1 | 1 |
| Errors in generated advisories | 0 | 0 |

As shown in Table 2, we classify two types of accountings for deviating from 'ground truth': *misses* are non-deliberate errors when an advisory should have been generated according to the operating rules; *omissions* are deliberate decisions not to generate an advisory.

TABLE 2
SCENARIO 1 MISSES AND OMISSIONS

| | **Misses** | | **Omissions** | |
|---|---|---|---|---|
| **Cause** | **Human** | **Auto ATS** | **Human** | **Auto ATS** |
| Perception | 1 | 0 | 0 | 0 |
| Geometry | 0 | 1 | 0 | 0 |
| Knowledge | 0 | 0 | 0 | 0 |
| Unknown | 0 | 0 | 0 | 0 |

In the case of Scenario 1, the same miss occurred for both the human and the AutoATS, though for different reasons. The human controller did not see the conflict because he had scrolled to another area of the screen. In the case of AutoATS, there was a difference in size of one corridor between how the human's system and the AutoATS, resulting in the miss. Essentially, this is an inconsistency between the ASM Console and AutoATS corridor data.

Tables 3, 4 and 5 below summarize the ASM/human versus AutoATS results for Scenario 2 in detecting conflicts and generating advisories. Scenario 2 included much more variation in the behavior of the air entities, thus more events.

TABLE 3
SCENARIO 2 RESULTS: HUMAN VS. AUTOATS

| | **ASM/ Human Controller** | **AutoATS** |
|---|---|---|
| Events | 25 | 24 |
| Advisories Generated | 21 | 21 |
| Advisory Difference | 4 | 3 |
| Errors in advisories | 2 | 8 |

TABLE 4
SCENARIO 2 MISSES AND OMISSIONS

| Cause | Misses | | Omissions | |
|---|---|---|---|---|
| | Human | Auto ATS | Human | Auto ATS |
| Perception | 3 | 0 | 1 | 0 |
| Geometry | 0 | 1 | 0 | 0 |
| Knowledge | 0 | 0 | 0 | 4 |
| Unknown | 0 | 0 | 0 | 0 |

In the case of Scenario 2, there were a number of misses and errors as shown in Table 4. The human controller had a number of perceptual misses where he was simply looking elsewhere in the battlespace, and did not catch the conflict. In one case (the human perception omission), the human operator saw the conflict but did not respond in time before the situation resolved itself (aircraft flew back into corridor). As with Scenario 1, in the case of AutoATS, there was a difference in size of one corridor between how the human's system and the AutoATS, resulting in the one geometry miss. The four knowledge-based omissions by AutoATS were due to particular rules that made it hold off generating an advisory if another advisory had already been issued to the aircraft in the recent past (60 seconds). This might occur if an aircraft had gone outside a corridor laterally then had an altitude violation in the same corridor – only one advisory would be generated. The human controller generated distinct advisories for both conditions.

TABLE 5
TYPES OF ERRORS IN ADVISORY GENERATION

| Error Type | Human | Auto ATS |
|---|---|---|
| Extraneous Advisory | 0 | 1 |
| Wrong ACM in Advisory | 1 | 3 |
| No ACM in Advisory | 0 | 4 |
| Wrong Flight Group | 1 | 0 |

It was only in Scenario 2 that we saw errors in the advisories that were generated, as shown in Table 5. In the case of the human operator, two such errors were made, and in the case of AutoATS, 8 such errors. In the human case, we can possibly attribute these errors to the increased burden in Scenario 2 – multiple conflicts would be detected at the same time, putting some stress on the controller to handle the next conflict. A few of the data points from AutoATS have not yet been accounted for, and for now can be attributed to a bug in programming. However, a few seem to be due to the aircraft flying out of one corridor near the intersection of another, but has not yet corrected back into the next one, so the advisory refers to the prior corridor. In the case of the human controller, these situations result in a correct advisory, and seem to be based on assumptions about the intent of the aircraft and knowledge about what the next new corridor should be. To account for this kind of reasoning would require more knowledge within the AutoATS agent to generate the right advisory, based on a few kinds of knowledge that it does not currently use.

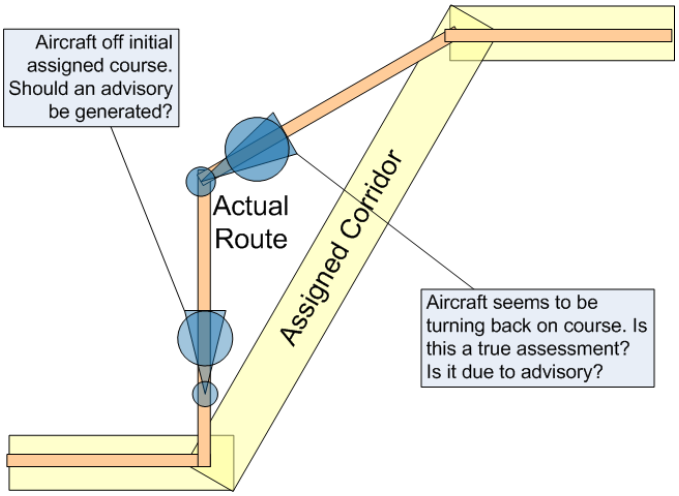


Fig. 2. Knowledge Use in Conflict Detection
(Blue icon is a helicopter moving north/northeast)

Advisories generated by the human controller were generally more forgiving than those derived algorithmically by AutoATS. For example, if the human controller saw the aircraft moving back toward the correct corridor, he would not issue a further advisory. AutoATS did not use this kind of inferred information. In a few cases, an aircraft deviated from its assigned corridor then returned to compliance before the human controller could generate an advisory. Strategy accounts for some of the remaining differences, such as with the re-generation of advisories for the same infraction. In some cases, the human controller would let the same infraction slide for a long period of time; AutoATS used a fixed 60-second window to make another request. Reaction time is also of some interest. The time for the human controller to generate an advisory averaged 16 seconds and 34 seconds for Scenarios 1 and 2, respectively. Some of this is accounted for in visual search of the ASM console, and searching the display for information to construct the appropriate advisory. AutoATS reacts immediately.

Many of the differences can be placed into the category of expectations and intent recognition. Once the controller issues an advisory, built-in expectations about the subsequent behavior of the aircraft guide further interactions. For example, if the controller tells the pilot he is off course, the controller gives the pilot some time to get back on course before issuing another advisory. If the controller judges that the pilot intends (by verbal response and visible behavior) to comply with the controller's request, then the controller need not issue another advisory until the aircraft appears to be breaking request. An example is illustrated below in Fig. 4. Intent estimation and recognition play a central role in matching a controller's expectations to reality and finding where adjustments must be made, either in the controller's mental model of the situation, or in the aircraft's actions.

## VI. CONCLUSION

We have described a system that automates aspects of airspace management, modeling an Air Traffic Services task that to date has been performed by a human in a simulation

environment. The results of the system are preliminary, given the limited scope of the experiment, but they point to the potential efficacy of automating these behaviors. The results have generated hypotheses regarding the kinds of knowledge required to effectively perform this task and indicate where we need further experimentation to test these hypotheses. Other capabilities of the system have not been evaluated, namely the quality, understandability, and correctness of the advisories that are generated by the system.

The capabilities described here – detecting and resolving airspace conflicts – are only a small slice of the capabilities required for full automation of Air Traffic Services or Air Traffic Control. There are obvious applications for a more complete system in training aviators or controllers. The limited capabilities might even be useful for on-board pilot assistance in navigation. In addition to automation or assistance, we can imagine using this tool for pre-mission analysis such as finding where pre-planned routes might conflict.

## VII. Relation to Prior Work

A good deal of attention has been paid to automating aspects of air traffic control, including analyzing the impact on workload [9], the SA requirements of the task [6, 10], and the development of cognitive models to evaluate task performance under varying conditions [11]. Other modeling work has examined alternative airspace policies [12], planning and scheduling algorithms for traffic flow [13], etc. Our work here is aimed at a broad but practical implementation of an interactive controller that can engage human or synthetic pilots in positive control, for the purposes of experimentation or training in human-in-the-loop simulations.

## Acknowledgments

Many thanks to the entire Soar Technology team for making this work possible, to AMRDEC for supporting this work and providing us a means to evaluate our effort to date, and to our volunteer human controller for putting up with our data collection requests.

## References

[1] JP3-52, "Joint Pub -- Doctrine for Joint Airspace Control in a Combat Zone," 2002.

[2] R. M. Jones and R. E. Wray, "Comparative Analysis of Frameworks for Knowledge-Intensive Intelligent Agents," *AI Magazine*, vol. 27, pp. 57-70, 2006.

[3] M. R. Endsley and D. G. Jones, "Situation awareness in air traffic control," presented at First International Conference on Applied Ergonomics, Instanbul, Turkey, 1996.

[4] M. R. Endsley, "Design and Evaluation for Situation Awareness Enhancement," presented at Human Factors 32nd Annual Meeting, Santa Monica, CA, 1988.

[5] M. R. Endsley and M. D. Rodgers, "Situation Awareness Information Requirements for En Route Air Traffic Control," Us Department of Transportation DOT/FAA/AM-94/27, 1994.

[6] M. R. Endsley and D. G. Jones, "Situation awareness requirements analysis for TRACON air traffic control," Texas Tech University, Lubbock, TX TTU-IE-95-01, 1995.

[7] G. Taylor, J. Miller, and J. Maddox, "Automating Simulation-Based Air Traffic Control," presented at IITSEC, Orlando, FL, 2005.

[8] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An architecture for general intelligence," *Artificial Intelligence*, vol. 33, pp. 1-64, 1987.

[9] C. D. Wickens, *Engineering Psychology and Human Performance*. New York, NY: Harper Collins, 1992.

[10] M. R. Endsley and M. D. Rodgers, "Situation awareness information requirements for en route air traffic control," presented at Human Factors and Ergonomics Society 38th Annual Meeting, Santa Monica, CA, 1994.

[11] K. A. Gluck and R. Pew, "Modeling Human Behavior with Integrated Cognitive Architectures: Comparison, Evaluation," Lawrence Erlbaum Associates, 2006.

[12] K. A. Harper, S. S. Mulgund, S. L. Guarino, A. V. Mehta, and G. L. Zacharias, "Agent-Based Performance Assessment Tool for General Aviation Operations Under Free Flight," presented at AIAA Guidance, Navigation and Control, Portland, OR, 1999.

[13] W. C. Meilander, M. Jin, and J. W. Baker, "Tractable Real-Time Air Traffic Control Automation," presented at 14th IASTED International Conference on Parallel and Distributed Computing and Systems, 2002.