An Intelligent Synthetic Wingman for Army Rotary Wing Aircraft

Randolph M. Jones, Alan J. Wallace, Jens Wessling
Soar Technology, Inc.
Ann Arbor, MI
rjones@soartech.com, wallace@soartech.com, wessling@soartech.com

ABSTRACT

The Army has a rich store of highly immersive flight simulations/simulators. Due to the expense of deploying multiple flight simulators, they are often used in experimental scenarios that only represent a single aircraft at a time. However, this is unrealistic because modern tactical Army aviation rarely flies solo, rather flying at a minimum in pairs. To enable more realistic simulation while reducing costs, applications often use constructive simulation of entities. However, the standard implementations of constructive entities sacrifices simulation fidelity by using low-cost desktop simulations that do not provide the precision and accuracy necessary in modern simulated warfighting exercises. A desirable solution would decrease cost while also retaining realism by providing autonomous, tactically correct, high-fidelity behaviors for the constructive simulated entities. This is the goal being addressed by the Automated Wingman project. This project integrates a state-of-the-art simulation architecture with the most advanced current technology for building knowledge-intensive intelligent agents. In addition to the most immediate application, providing automated wingmen for Army experimentation with rotary-wing aircraft, this project provides a more general opportunity to broaden the use of Intelligent Synthetic Force (ISF) models in DoD applications. The industrial-strength integration of the Soar architecture for intelligence and the VR-Forces simulation environment creates a robust platform for future applications both in the DoD and the commercial arena. This integration relies upon a clean design that includes independent but interacting components. As a consequence, the resulting system contains individual parts that can be reused or upgraded as future demand and development dictate.

ABOUT THE AUTHORS

Randolph M. Jones, Chief Scientist and Vice President, Soar Technology, Inc., earned his Ph.D. in 1989 from the Department of Information and Computer Science at the University of California, Irvine. He holds positions as a Visiting Assistant Professor at Colby College and Visiting Assistant Research Scientist at the University of Michigan, and he has previously held research positions at Carnegie Mellon University and the University of Pittsburgh. His general areas of research include computational models of human learning and problem solving, and automated intelligent actors. He has worked on the TacAir-Soar project since its inception, and wrote the first implementation of the TacAir-Soar system. He is actively involved in Soar Technology's Lockheed Martin/Tactical Defense Systems contract for design, development, testing and documentation of the AMN/Link4-A training capability within the Battle Force Tactical Trainer (BFTT), adding behaviors for landing aircraft on ships and creating agents dynamically.

Mr. Al Wallace, Chief Operating Officer, Soar Technology, Inc., joined Soar Technology in 2001 after more than 13 years with the Air Force in the Engineering Directorate's Modeling and Simulation Division at Aeronautical Systems Center (ASC), Wright-Patterson Air Force Base, Ohio. While he was with the Air Force, Mr. Wallace held a variety of positions including software engineer, model/study and program manager for the Logistics Composite Model (LCOM), as well as Technical Expert for the Modeling and Simulation Division. Mr. Wallace supported a number of different weapon system programs, including the Joint Strike Fighter (JSF), F-22 Raptor, C-17, B-2, B-1B, F-15 Eagle, F-16, Airborne Laser (ABL) and others. He has also supported a variety of U.S. Air Force, Marine, Navy and Special Operations Forces' activities. His current responsibilities include monitoring and analysis of company operations, management of personnel, and project management for Internal Research and Development (IR&D). In addition to his COO duties at Soar Technology, Mr. Wallace acts as Program Manager for a variety of government-funded projects, including the Automated Wingman Project. He also has a special

interest in the Validation & Verification (V&V) of intelligent agents. He received his BS in Computer Engineering from the University of Michigan, Ann Arbor (1987) and his MS in Engineering Management from the University of Dayton (2000).

An Intelligent Synthetic Wingman for Army Rotary Wing Aircraft

Randolph M. Jones, Alan J. Wallace, Jens Wessling
Soar Technology, Inc.
Ann Arbor, MI
rjones@soartech.com, wallace@soartech.com, wessling@soartech.com

INTELLIGENT CONSRUCTIVE FORCES

As in much of the DOD, the US Army relies heavily on simulated environments for training and experimentation. For experimental purposes, the Army maintains a rich store of highly immersive flight simulators and simulated environments. Most of these immersive systems provide high fidelity simulation of a single aircraft, isolated from the multi-player realities of operational environments. This fact creates obstacles to training and experimentation, because modern military aviation tactics rarely involve solo missions. Rather, aircraft support each other in groups in teams, at a minimum in pairs. To provide realistic simulations of such systems, one solution is to augment the simulation environment with additional high-fidelity flight simulators, together with human role players to control them.

However, this approach is prohibitively expensive for many experimental situations. Constructive simulation of multi-player teams has the potential to provide a lower cost alternative. In constructive simulation, computer software generates the representations and actions of individual entities, often under the direction of a human operator sitting at a control panel. However, the use of constructive forces comes with at least two types of costs. First, the user interface for the human operator is necessarily much lower fidelity than a full flight simulator, which can impact the reality of the simulation. Second, this type of simulation requires the use of human operators who are generally trained both in operations (e.g., aircraft pilots) and trained in how to operate the constructive forces. While cheaper than using a full suite of virtual simulators, the need for the specially trained personnel can be expensive.

A fruitful alternative is to use knowledge-intensive agent technology to drive the constructive forces. Using knowledge-based agents increases the autonomy of the simulated forces. Appropriate levels of autonomy have the potential to produce human-like realism in the behaviors while simultaneously reducing the manpower requirements for running the simulation.

Helo-Soar

This report describes initial work on Helo-Soar, a knowledge-intensive intelligent agent system designed to pilot constructive simulations of rotary wing aircraft for Army experimentation. The ultimate goal of Helo-Soar is to provide an "Automated Wingman" the flies in teams with human pilots in virtual flight simulators. To accomplish this, Helo-Soar must encode human-like knowledge for a variety of tasks. To begin with, Helo-
Soar must encode doctrinal knowledge and competence for performing particular types of rotary-wing missions. In addition, because the main application involves teams of constructive forces and humans, Helo-Soar must be able to reason about how to coordinate in groups, following appropriate command structures and lines of communication. Finally, Helo-Soar must include knowledge of how to interact and communicate with its teammates. This includes the ability to recognize and generate speech, to parse speech into meaningful directives, and to engage in structured conversation following appropriate communications cadences.

All of these requirements provide significant challenges to building a successful system. The knowledge engineering task itself is daunting. However, we are building on the success of TacAir-Soar (Jones et al., 1999), an existing system that flies simulated fixed-wing aircraft, as well as prior work building intelligent agents for rotary-wing aircraft (reference BFTT and maybe USC). In addition to the knowledge engineering task, Helo-Soar must have access to appropriate realistic models of the physical flight platforms. This also implies the design of a targeted level of interface for agent control of the simulated entities. Helo-Soar must also integrate smoothly with external systems for speech synthesis and recognition. Finally, the system must be embedded in a larger distributed simulation architecture that is robust and flexible.

Component orientation

The remainder of this report describes Helo-Soar and the architecture within which it is embedded. The guiding design principles for the entire system rely on a component oriented approach. The basic simulation architecture is MaK Technologies' VR-Forces, which provides an extremely modular and well-designed set of components for integrating distributed simulations. In addition to providing the simulated environment, VR-Forces provides the physical flight models that Helo-Soar controls. An additional module provides the "human level" interface between the physical models and the intelligent agent architecture. Following the component-oriented paradigm, the intelligent agent architecture and speech modules also serve as distinct modules within the overall system. This component orientation maximizes flexibility and the potential for reuse, as well as future incremental improvements of the various components in the system. We have also carried the component-oriented paradigm inside the intelligent agent's design, where knowledge is organized around a modular set of interacting goals that provide efficient but goal-driven reasoning.

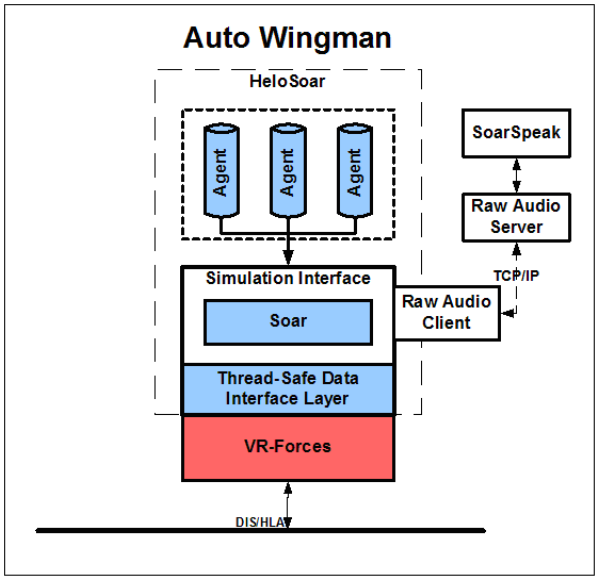

Figure 1. Automated Wingman System Architecture

AUTOMATED WINGMAN SYSTEM ARCHITECTURE

Helo-Soar is one piece of a larger architecture for distributed simulation. A schematic of the architecture as a whole appears in Figure 1. The Auto Wingman system architecture is built to allow Soar agents to interface with VR-Forces and to control entities within VR-Forces. The overall system is designed to minimize direct interactions between low-level Soar interface code and low level VR-Forces interface code. It does this by providing a Thread-safe data interface layer.

As each Soar agent is created, it is bound to a specific construct on the simulation side that is responsible for managing it's interaction with VR-Forces. This thread-safe binding provides the interface layer that moves data and commands back and forth between the agent and VR-Forces as XML.

The state of the world in the simulation is collected in an XML document each simulation cycle and the document is passed to the agent and directly placed in the agent's working memory. This conversion is agnostic to the form of the XML coming in. This allows the simulation to add additional information to the input link of the agent without actually requiring any changes to the low-level Soar interface code. Because Soar agents ignore any data that they are not specifically looking for there is a minimal penalty for sending extra data to the agent that it may, or may not, need in the future. In the event of multiple updates to the world state between agent input phases, the most current world state is used.

Commands from the agent are taken from the agent's output-link and directly converted to XML that can be processed by the simulation. These commands are well documented and can be validated using either an XML schema, or a DTD. This insures that the commands coming from the agent are not only well formed, but valid commands. After the command is received, it is processed and the action is taken. In the event that the agent sends multiple commands between VR-Forces updates, the commands are queued and executed at VR-Forces' earliest convenience.

This architecture design has proven flexible and robust in providing a very capable agent based interface between Soar and VR-Forces.

INTELLIGENT AGENT REPRESENTATION

One of the biggest challenges to building the Automated Wingman system is to engineer the intelligent agent that controls the RWA platforms and interacts with human commanders and pilots. To build Helo-Soar, we are relying on an enhanced version of the methodology we have previously used in building TacAir-Soar (Jones et al., 1999), a similar system that controls simulated fixed-wing aircraft.

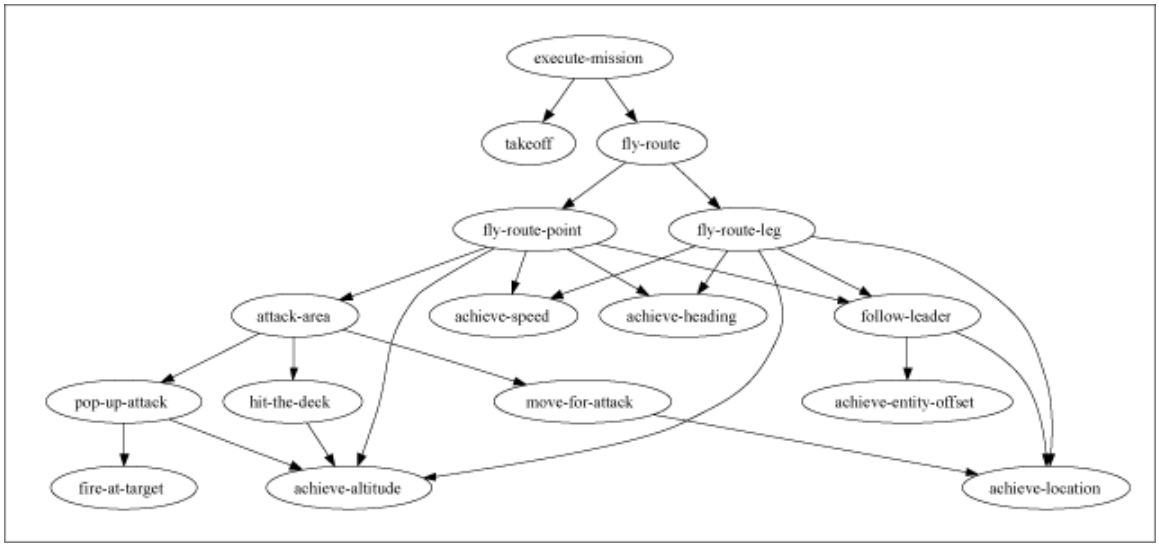

Figure 2. A directed acyclic graph of interacting goals to direct Helo-Soar's behavior.

Both of these systems are designed within the Soar architecture (Laird, Newell, & Rosenbloom, 1987; Newell, 1990), which provides an efficient engine for executing production-rule agents. In Soar, all long-term knowledge must ultimately be engineered into a set of production rules, which are efficient relational pattern matchers that provide associate retrieval of knowledge to drive intelligent behavior. In the Helo-Soar system, we divide these productions into three broad categories. The first maintains the (potentially intricate) network of interacting goals that the agent must attend to at various times, the second implements interpretation knowledge that builds an elaborate representation of situational awareness from the agent's perception and interpretations of the environment, and the third implements discrete deliberate actions that the agent must take in service of its goals, and in the context of its representation of situational awareness.

Figure 2 shows a schematic representation of a subset of the goals in the Helo-Soar system. The main point to glean from this diagram is that there is a hierarchical organization of potentially interacting goals that can be active at various times. The hierarchical arrangement provides a tiered set of modular contexts for the agent's reasoning. This can be important in implementing the efficient retrieval of knowledge within the agent. Recall that long-term knowledge in Soar is implemented is triggered by a set of relational pattern matchers. Patterns appropriate to the *attack-area* goal can be monitored independently and continuously even as subgoals of *attack-area* change with the dynamic situation. For example, there are certain types of situation interpretations (for example, dealing with target location) that the agent must perform during any area attack, but other interpretations are only relevant for particular subgoals (for example, low-level maneuvering for the *move-for-attack* subgoal).

Additionally, this hierarchical organization allows the knowledge engineers to impose high-level structure and encapsulation on the knowledge base. The ability to do this is essential for large, knowledge-based systems. Without such high-level organization, as with traditional large software projects, the agent system can become prohibitively expensive to maintain and adapt to new requirements. This is particularly true for intelligent systems, which by their very nature need to reason about the interactions between goals, and complex relational patterns between goals and situational interpretations.

It is also this relational intertwining of knowledge that limits the ability to encapsulate knowledge structures completely. Although the knowledge design attempts as much as possible to define a clean hierarchy of goals and subgoals, it should be clear from Figure 2 that a simple tree structure is not sufficient to support intelligent behavior. Many subgoals can be performed in the service of a variety of supergoals, potentially causing problems for encapsulated behavior. In addition, deliberate actions generated by the agent may have to switch quickly between attending to multiple active goals, or take multiple active goals into account simultaneously. A good example has to do with the particular role of an automated wingman. A wingman for a mission must monitor and execute the overall

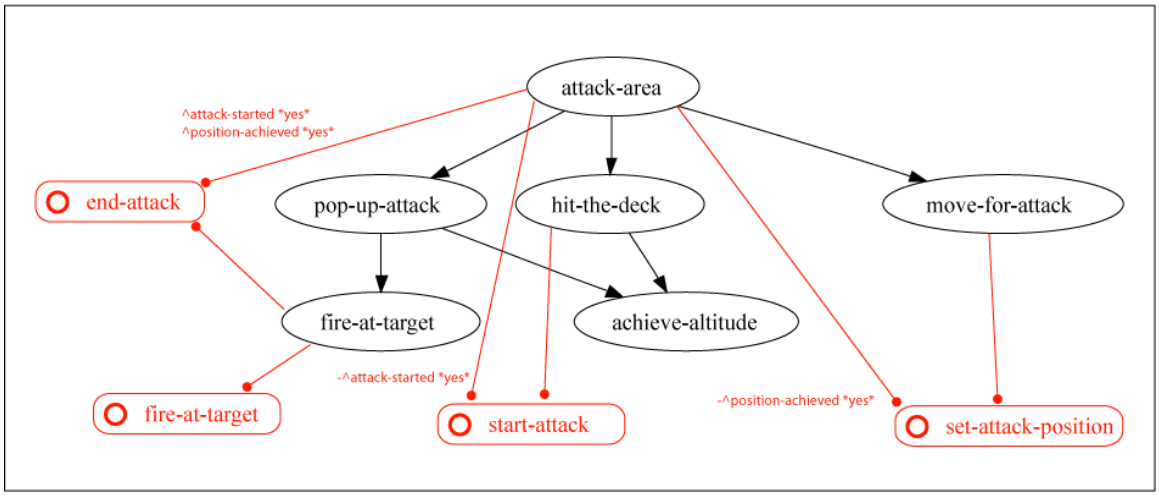

Figure 3. A subset of Helo-Soar's goal hierarchy, including a subset of potential discrete actions relevant to the goals.

mission along with the rest of its teammates. So an automated wingman, for example, may be attempting to achieve the *attack-area* goal together with the rest of the group. Simultaneously, however, a wingman must focus on the special requirements of providing various support actions for the lead, as well as maneuvering to stay in formation. The *follow-leader* goal explicitly represent these constraints in Figure 2. For simplicity, we have omitted some of the explicit subgoals of *follow-leader*.

The point here is that an intelligent automated wingman must interleave actions in support of the *attack-area* and *follow-leader* (and probably also other) goals simultaneously. Our knowledge representation approach in Helo-Soar makes these simultaneous interactions explicit, allowing the potential for meta-reasoning across interacting goals, and allowing multiple independent (or dependent) goals to drive reasoning, situation interpretation, and action.

This knowledge design also takes direct advantage of the truth-maintenance system (Forbus & deKleer, 1993)built into the Soar architecture. A truth maintenance system automatically computes logical entailments of an agent's perceptions, assumptions, and other contextual features. In the context of the goal hierarchies, the truth maintenance system automatically maintains the dependencies between goals, both for goal-subgoal relationships, and for connections between goals that interact. As the environment changes, or the agent changes its interpretation of the environment, the truth maintenance system rapidly and automatically restructures the active goal structures to remain consistent with the situation. This allows the agent to generate appropriate deliberate actions with confidence, even in the face of a rapidly changing, complex environment.

Helo-Soar also uses Soar's truth maintenance system to maintain a representation of situational awareness. As inputs to the agent change (from changes in the environment, via Soar's interface to VR-Forces), the truth maintenance system automatically computes logical entailments of those changes. For example, a pop-up contact on the simulated radar might combine with existing knowledge about friendly and enemy locations to infer a new threat. Consequently the existence of a threat could combine with knowledge about the agent's flight parameters and the threat's maneuvers to produce various geometric and spatial interpretations of the situation. Additional knowledge can use the spatial representations to suggest appropriate weapons to use, or ultimately new goals to activate (such as evasive maneuvers or air-to-air attacks). Given the appropriate set of productions encoding knowledge, all of these interpretations can change rapidly and automatically to remain consistent with changes in the environment.

Both the goal structures and situational representations have to do with maintaining the agent's internal state of awareness. But ultimately an intelligent agent must take some form of external action in the environment. The Soar architecture provides this type of mechanism by supporting productions that implement deliberative, discrete actions, in response to the current context provided by active goals and beliefs. In Soar, a deliberative, step-wise action is called an *operator*. During each discrete time-step of the simulation, Helo-

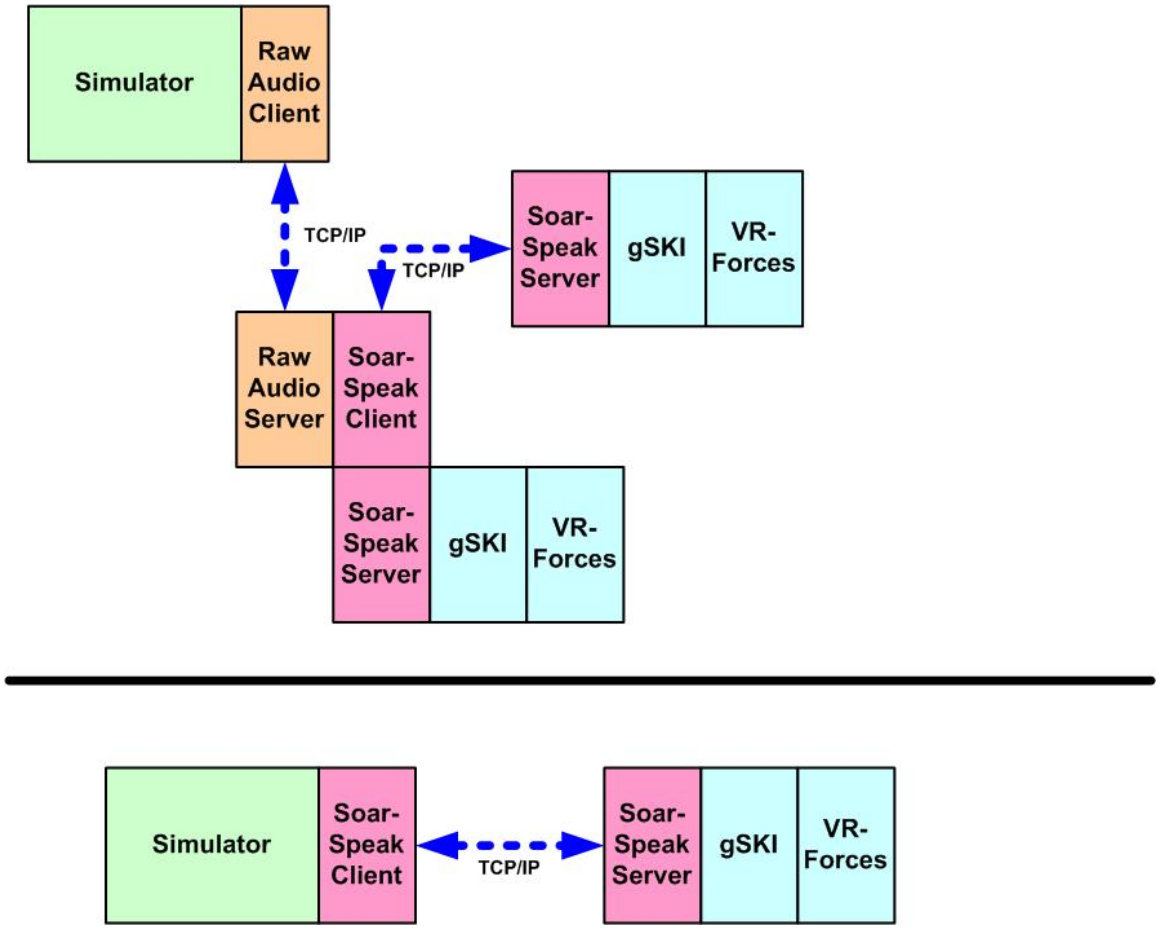

Figure 4. Integration of SoarSpeak speech recognition and generation components into the simulation architecture.

Soar determines the next appropriate operator to select and execute (often this operator may simply be to *wait*, if Helo-Soar needs some change in the environment to drive further action).

Figure 3 shows some of the operators associated with a subset of the goals encoded into Helo-Soar. The bottom layer of elements in the figure represent discrete operators that Helo-Soar will take in a context-dependent manner. Selection and execution of these operators depends on the activation of the connected goals, as well as various situational features than can be attached to the goal structures. As a simple example, the operators *start-attack*, *set-attack-position*, and *end-attack* can all be selected in the context of the *attack-area* goal. However, each also depends on other subgoals being active, as well as particular values of associated features with the *attack-area* goal. Among other roles, the *attack-area* goal structure maintains a record of the current status of the attack, including whether the attack has started and whether the attack position has been achieved. At various stages of the attack, and in response to the activation of various other subgoals, appropriate operators will implement the procedures for beginning the attack, moving to the appropriate location, and ending the attack. Similarly, the agent will invoke the deliberate behavior to fire at a target (by explicitly invoking the missile model provided by the simulation architecture) in an appropriate context depending in part on the activation of the *fire-at-target* goal.

Keep in mind that all of these actions are discrete, and will interleave with other actions in the service of other goals. For example, while attempting to achieve the *attack-area* goal, Helo-Soar will also be generating deliberate actions to achieve communication goals, fly in formation, maintain situational awareness, evade threats, and other activities. This interleaving (together with the knowledge management techniques for building and maintaining this knowledge-rich system) is made possible by the modular goal representations with explicitly defined interactions, which is supported

in large part by the automated logical consistency provided by Soar.

HUMAN-AGENT COMMUNICATION

Human-agent communication is handled through the SoarSpeak subsystem, a speech recognition and generation interface for Soar agents. SoarSpeak provides interfaces to translate spoken directives into English text, based either upon a socket-based protocol, or upon the High Level Architecture (HLA). This text is sent across simulated radio channels where it is "understood" and acted upon by the simulated wingman. For speech generation the reverse process occurs; the simulated wingman generates a text message that is sent across a simulated radio to a speech synthesis application for the human pilot to hear.

In developing Helo-Soar, it was necessary to integrate SoarSpeak not only with the simulation system but also with the simulator, itself. The desire to minimize computing and footprint requirements for the simulator platform lead to the design capture in Figure 4. In this implementation, a Raw Audio Client/Server application was developed to allow raw audio to by captured, packaged up and shipped across the network to the SoarSpeak client-server. The raw audio client can then reside on a computer that is co-located with the simulator, while the raw audio server can reside on the same platform that the simulation is run on. Connections are handled via TCP/IP sockets.

In order to achieve acceptable voice recognition rates, a restricted grammar was developed and is the means used to first recognize the pilot's utterance, and then to be "understood" by the intelligent agent wingmen. The basic operation of the system is as follows: the pilot in the simulator speaks into his microphone and issues orders as he would normally issue to another human pilot. The intelligent agent recognizes and understands the orders and replies using the voice generation system according to military doctrine and then goes ahead and executes the orders it has received.

INTEGRATION LESSONS

One of the key aspects of this effort is the integration of Soar with the VR-Forces simulation architecture. This integration provides one of the most sophisticated cognitive architectures (for implementing intelligent agents) with one of the most modern and robustly engineered simulation platforms (for providing physical and environmental simulation). Integrating Soar with VR-Forces was unique in several respects. This is the first time Soar has been integrated with a heavily object oriented simulation. This is in contrast to other simulations with which Soar has previously been integrated, such as JSAF and OTB. This is also the first time Soar has been configured to run asynchronously with a sophisticated simulation environment. In the Automated Wingman architecture, the Soar component runs in its own thread, while the rest of the simulation runs in another. As a final constraint, this is the first significant Soar integration where the integration was performed at the binary code level; the Soar engineers did not have direct access to the VR-Forces source code to support the integration of the intelligent agents. This was a particularly important test of the benefits of a good modular design for the overall system architecture.

While integrating with an object oriented system is much cleaner, it also requires a very careful understanding of the overall simulation architecture. For this project, it has been critical for the Soar architecture developers to work closely with the simulation developers, in order to understand the design concepts necessary to accomplish the integration. This "front-loaded" the integration effort. However, once the initial design was created, the implementation was straightforward.

The requirement for asynchronous control has also been an issue. When integrating with multiple distributed simulations, it is important to stay as synchronized as possible, so no component of the distributed system gets too much out of synch with the others. Historically, intelligent agents have been integrated to work in lock step with the underlying simulation platforms. This made it impossible for the simulation to get out of synch with the agents (in particular, providing some control over the required reaction times of the agents). By dissociating the simulation from the agents, if either is idling the other thread can make use of the additional cycles. For example, if the simulation is "sleeping", waiting for data, the agent can continue to process and reason over the data that is so far available. This is an advantage to the type of "anytime" reasoning that these intelligent agents engage in.

The commitment to keeping a clean separation between source code changes to Soar and to the simulation engine was a potentially high risk design decision. This effort could have been a major catastrophe if not for the effective and expeditious support of the simulation developer. Because agent control over simulation entities intimately integrates

the simulation with the agents, it is important that the details of the simulation and the agent architecture are well defined, and well understood by all groups in the development team. Because intelligent agents have some fundamental differences from standard constructive simulations, it would be unusual to discover that no changes are required at all to the underlying simulation. However, we were able to tame this process through well-defined interfaces and the close cooperation of the simulation development team.

CONCLUSION

We have described a simulation architecture to support the use of intelligent automated wingmen for army experimentation with rotary-wing aircraft. The design includes the integration of a well-designed set of components providing the distributed simulation environment, physical platforms, intelligent agents, and various interfaces for agent communication with the simulation, with other agents, and with humans. In addition, the intelligent agents implements a new modular approach to knowledge-intensive agent design, exploiting an efficient low-level architecture for intelligence, and organizing high-level knowledge around complex representation of interacting goals, efficient representation of sophisticated situation interpretations, and the uniform integration of deliberate action that is appropriately reactive to changing situation interpretations, as well as sensitive to multiple active goals. This component-oriented design provides a robust and well engineered architecture for DOD applications, which will provide a solid software engineering solution for Army experimentation, while also reducing maintenance and upgrade costs for future expansion of the system to new application areas.

REFERENCES

Forbus, K., & deKleer, J. (1993). *Building Problem Solvers*. Cambridge, MA: MIT Press.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P. G., & Koss, F. V. (1999). Automated Intelligent Pilots for Combat Flight Simulation. *AI Magazine, 20*(1), 27-42.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*(3), 1-64.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, Massachusetts: Harvard University Press.