# CIANC[3]: An Agent-Based Intelligent Interface for Future Combat Systems Command and Control

*Scott Wood*
*Jack Zaientz*
*Jonathon Beard*
*Richard Frederiksen*
Soar Technology, Inc.
3600 Green Court Suite 600
Ann Arbor, MI 48105
734-327-8000
swood@soartech.com, jzaientz@soartech.com, beard@soartech.com, rdf@soartech.com

*Marcus Huber*
Intelligent Reasoning Systems
4976 Lassen Dr.
Oceanside, CA 92056
760-806-1497
marcush@marcush.net

Keywords:
Intelligent Agents, Intelligent User Interfaces, Usability, Interoperability and Organizational Behavior Representation

***ABSTRACT:*** *The vision of the future for armored and mechanized military structure, as spelled out by (DARPA, 2001), includes the use of mixed teams of human and robotic forces on a dynamic and rapidly changing battlefield. Successful implementation of this shift will require autonomous and semi-autonomous robotic forces and a command and control infrastructure that will allow human, robotic, and mixed teams to be controlled quickly and easily. This infrastructure will need to allow human commanders to control the robot teams in a similar manner to how they command human teams, that is, in the language of the military, not the language of robotic control theory. In this paper, we present an intelligent command and control interface built on framework of cooperative interface agents that reflect roles found in current command staffs.*

## 1. Introduction

The vision of the Army's Future Combat Systems (FCS) includes the use of mixed teams of human and robotic forces on a dynamic and rapidly changing battlefield. Implementing the vision will require a shift from manual, human control of weapons systems to semi- and fully autonomous control over mixed systems of humans and non-human entities. It will also entail an overall force reduction that will require multiple entities to be controlled by individual team leaders and multiple teams to be lead by higher-echelon commanders.

To accomplish this, systems will have to be designed to require less human interaction and greater robotic autonomy. Successful implementation of this shift will require autonomous and semi-autonomous robotic forces and a command and control infrastructure that will allow human, robotic, and mixed teams to be controlled quickly and easily. One key to this will be the degree to which teams and individual robots are autonomous. A second is whether the commander's human-machine interface is designed such that the commander is not overloaded with constant system interaction allowing him or her to focus on the mission.

The focus of this project (CIANC[3]: Cooperative Interface Agents for Networked Command, Control and Communications) has been to identify the human-interface issues, design potential solutions

and create intelligent agent software that support the commander's tasks and mitigates human performance limitations in the context of robotic command and control. To accomplish this task we have implemented an agent architecture based on decomposing the command and control problem into three main task areas: Monitoring, Coordinating and Tasking. By using agents that specialize in each of these three areas as an interface to the underlying robotic behaviors (represented by Task Frame entities), we have been able to rapidly develop an intelligent interface for the robotic entities that make up FCS.

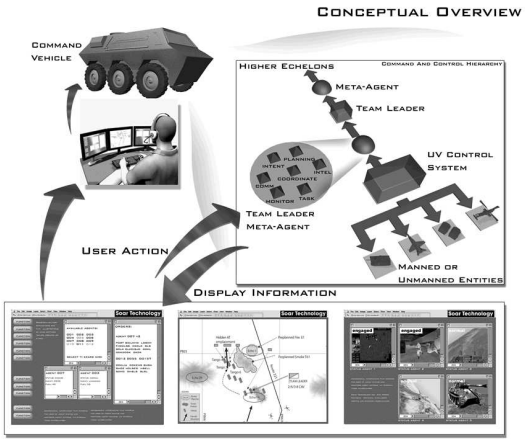


**Figure 1: Conceptual overview**

### 1.1 Robotic Battlefield Entities

An overall goal of the FCS program is to transform the current military structure, operations, strategies and tactics to create a force that is more responsive, deployable, agile, versatile, lethal, survivable, and sustainable. One implementation strategy to achieve this goal is to split the roles of battlefield entities to create smaller, more specialized platforms that will operate cooperatively in a much more effective manner than currently possible. This will include at least the following battlefield platforms: manned vehicles, direct fire vehicles, indirect fire, beyond-line of sight (BLOS) vehicles, sensor vehicles, unmanned aerial vehicles, and other layered sensors such as satellites (c.f. Van Fosson, 2000; and Defense Advanced Research Projects Agency, 2000). Other research is addressing low-level issues regarding autonomous robot control, such as cooperative path planning, team selection and tactics, and dealing with uncertainty, e.g. the MICA project (Defense Advanced Research Projects Agency BAA #01-029).

The present work will develop software techniques and technologies that will allow human commanders to control the robot teams in a similar manner to how they command human teams, that is, in the language of the military, not the language of robotic control theory. In addition, it will address command and control for higher echelons and for cooperative actions across echelons.

### 1.2 Human-Machine Interaction and Supervisory Control

The overall goal of the human-machine interface design for this project is to maximize human performance by creating a system that allows users to perform military tasks without focusing on the computer system used. This requires a system that is efficient to use, easy to learn, easy to remember, and error-tolerant

One technique for maximizing usability is to automate mundane and time-consuming tasks with software. Previous efforts at automating system tasks have achieved mixed results often because supervisory control issues (Leveson, 1995; Sheridan, 2000) were not adequately addressed. Effectively automating system functions requires a delicate balance of reducing tedious tasks and overall operator workload, and maintaining adequate human control (both real and perceived) and vigilance. For example, users will become complacent in monitoring-only tasks, such as monitoring status gauges or security cameras, and become more prone to errors. They need to be kept engaged and they need to maintain their skills for times when automated systems are inadequate. Task-analytic techniques can be used to address the supervisory control problem, enabling designs that will include the right mix of human and automated control. One way of implementing supervisory control software is through software interface agents.

### 1.3 Interface Agents

Interface agents (Laurel, 1991) are a specific form of software designed to reduce the complexity of human-system interaction. Such agents can take the form of relatively simple agents for performing single, well-defined tasks such as filtering mail, or they can be fairly complex for more complicated tasks such as seeking out useful information or websites (Lieberman, 1997). Fundamentally, interface agents represent an additional, simplifying layer of abstraction between a user and a computer system.

Agents provide the interface with the capacity for a mixed-initiative dialog allowing for the more natural give and take characteristic of typical human conversation. Key elements of this dialog (Horvitz 1999) include the interface agent's ability to

- Consider uncertainty about the commander's goals.
- Consider the status of the commander's attention in the timing of services.
- Infer ideal action in light of costs, benefits and uncertainties.
- Employ dialog to resolve uncertainties.
- Allow direct invocation and termination of interface services.

This dialog between commander and system will provide a flexible level of control that can adapt to the dynamic environment of battlefield command, offering the commander as little or as much direct involvement as is required by situation, doctrine, or commander preference.

### 1.4 Multi-Agent Design

It is important that this interface technology be developed modularly, creating cohesive, loosely coupled entities that can be easily modified, adapted, and reconfigured as doctrine, technology, and missions evolve. Dividing agent workload between a set of specialized modular agent types provides a number of key benefits.

#### *14.1 Encapsulation of knowledge*

Localizing doctrinal knowledge in specialized agents provides a natural mechanism for matching interface-processing rules with military doctrine. CIANC$^3$ agents will be part of the DOD's C$^3$ structure, and as such will need to adapt to changes in doctrine over time and by service and operation. As requirements change, agents encapsulating the new rules can be introduced into the system without impacting on other aspects of the system.

#### *14.2 Encapsulation of Processing*

Localizing task execution in specialized agents also provides a natural mechanism for encapsulating processing. As the duties of the individual CIANC$^3$ agents increase in scope and sophistication, specialized techniques will be adopted or developed to increase task performance, robustness, or scalability. While our current research utilizes the Soar architecture for agent decision making, it is quite reasonable to assume that future CIANC$^3$ agents may require the addition of dedicated planners, case-based reasoning systems, or other AI technology.

#### *14.3 Communication oriented design*

It is also important to note that the division of knowledge and processing into distinct agent types creates a demand for a more sophisticated communication infrastructure than might be required by a monolithic system. This increased sophistication, despite the additional development requirements to construct it, is another one of the key benefits of the system. Since the purpose of the CIANC$^3$ system is assisting the commander in the command, control and *communication* of external robotic entities a rich capacity for communication is a basic requirement. Establishing this capacity as a fundamental characteristic of the architecture, available to individual processing units such as information display agents, allows the seamless introduction of new processing units at any time or at any location in the CIANC$^3$ architecture.

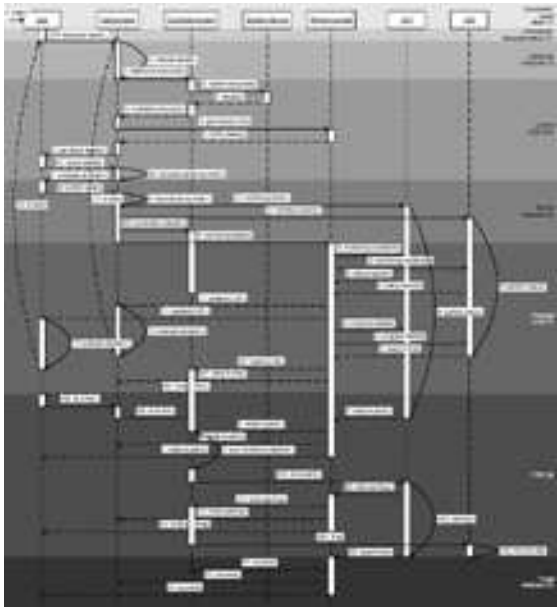

**Figure 2: UML Communication Sequence Diagram**

#### *14.4 Reconfigurable design*

It should also be assumed that the target agent organization described here would change to include other classes of interface agents. The agent architecture, therefore, must accommodate such change. For example, a display agent could be used to control all information presented to the user. An executive agent may be useful for coordinating the

control and communication within a collection of agents (e.g. within the meta-agent). Other agent roles that might be separately developed include:

- Deriving the commander's current task from recent actions.
- Deriving enemy intent based on recent enemy actions.
- Evaluating and critiquing plans.
- Routine scheduling of communications, supply, and duty rotations.

Additionally, the missions, roles, responsibilities and information requirements will be different for each echelon in which this technology is employed. Doctrine will also change with coming technological advances. It is important that the resulting system be flexible and modular enough to rapidly adapt to new procedures and protocols. For example, the agent system should be constructed to allow different sets of expert knowledge to be easily constructed and integrated into the agents. Tools for doing this are currently under development.

## 2. CIANC$^3$ Agents

We have created a framework of cooperative interface agents based on the roles found in current command staffs. Command staffs commonly provide five basic functions to commanders in support of reconnaissance, security, offensive, and defensive operations (c.f. Army FM 17-95):

- Provide timely and accurate information.
- Anticipate requirements and prepare estimates.
- Determine courses of action and make recommendations.
- Prepare plans and orders.
- Supervise execution of decisions.

In CIANC$^3$ these functions are divided between three classes of agents: tasking, monitoring, and coordinating. This division aligns the agents with the three C$^3$ concepts of command, control, and communication respectively. The goal is to provide a virtual command staff for echelons that do not current have that support. These command and control agents can then supervise a large set of more specialized agents, such as Intelligence, Security, Logistics tailored to the particular echelon. In addition, we see the need for platform oriented Display and Data channel agents to support data fusion and task specific visualization.

Figure 1 illustrates a notional organization for how the interface agents will work in the larger scheme of battlefield command and control. Although other configurations are possible, the basic roles and responsibilities required of the interface agents will remain. In addition, it is assumed that interface agents will have access to, and be integrated tightly with, other battlefield information and decision-support systems. Irrespective of the type of digitized services that will become available to battlefield commanders, the need for rapid tasking, coordinating, and monitoring of operations will remain. These agent classes are discussed below with examples of how they might be used.

### 2.1 Tasking Agents

Tasking agents will be used to assist commanders and controllers to rapidly issue battlefield commands. They are to reason about the commander's intent, standard operating procedures, unit capabilities, operating environment and enemy disposition to present the commander with a reasonable operation plan. Where ambiguity exists, tasking agents should engage the commander in dialog to clarify intentions or will present several options. After customizing the resulting plan as necessary, the commander can then issue the order. The tasking agent will then translate the order into the proper command sequences for next command layer. These sequences range from dialog completion information to atomic-level robotic commands, or relatively high-level commands that will be further processed by a cooperative planning system.

For example, a commander may wish to task a deployed company to attack a target. To do this he could select the company or individual platoon elements with a light pen (or other suitable input device) and drag them to the designated target area using the desired path and direction of attack. The tasking agent would then query the commander as to the mission type who in turn would select Attack. The agent would then reason about the current posture of the company, assets of the platoon elements, terrain, weather, and enemy, and propose a mission profile. An order would then be prepared specifying the commander's intent; movement orders indicating lead and screen elements, and other information normally included in an operation plan. After reviewing and verifying the plan, the commander would confirm the order; the tasking agent would translate the order (for robotic forces) and send out the plan. After confirming receipt of the order, the system would then monitor

the plan's progress and update the commander as necessary.

It is not enough that the system simply automate the commander's tasks. Users of the system must be aware of and feel in control of the situation at all times. Otherwise, they will either lose trust in the system, reverting to manual control, or place too much faith in it, becoming complacent and jeopardizing lives. After orders have been issued, the plans should be visible to the commander so that they can be inspected, monitored, critiqued, and modified. This mix of interface agent assistance and direct manipulation is essential to achieving the right mix of automated and manual control. Examples of other roles tasking agents might play include:

- Tasking UAVs for targeting.
- Automatic weapon selection for known target types.
- Automatically modifying defensive posture in the event of an ambush.
- Modifying weapons usage (rate of fire, ammo selection).
- Modifying alert rules for when an autonomous agent should seek guidance.
- Facilitate any direct manipulation of by providing context-sensitive assistance such as assigning targeting priorities.

### 2.2 Coordinating Agents

Coordinating agents are responsible for facilitating communication and coordination across and within echelons within the command hierarchy. While command hierarchies will certainly continue, operational hierarchies are likely to become more network-centric, blurring the distinction between separate commands. Units in one command may cooperate with a second command element one minute and a third the next. Such dynamic operational shifts will only be possible by automating much of the communication and coordination that must occur in such situations. Tasks such as determining radio frequencies, call signs, unit designations, chain-of-command, IFF and communications security are all time-consuming but necessary issues with which coordinating agents will be able to assist.

For example, coordinating agents can increase force lethality in cooperative engagements by minimizing duplication of effort, maximizing target coverage, synchronizing time of attack, or massing fire on a single target. They can also be responsible for maintaining a common operational picture (and thus, situational awareness) by updating higher and lower echelons on the current situation, plans, enemy intentions, and battle damage assessment. As with tasking agents, it is important that agent actions, processes, and results be visible to the user. The commander must be able to verify that his intentions are being accurately implemented, and he must be able to intercede when necessary.

Another example where coordination is critical is rapidly responding to fast-moving or stealthy targets. Coordinating air defenses and sensor systems faster than humanly possible is often necessary for effectively countering such attacks. In such situations, the coordination agent might work directly with monitoring and tasking agents to rapidly eliminate the threat. Other roles that might be played by coordination agents include:

- Setting up direct sensor to shooter communications across commands.
- Setting up other cross-command tasking such as indirect fire support.
- Facilitating teleconferencing.
- Reestablishing communications and integrating orphaned units.
- Communicating routes, plans, intentions, progress and other explicit and tacit information.
- Sharing incomplete sensor information (such as vectors to fire source) to higher echelons.
- Facilitating direct control of vehicles (e.g., tele-operation) in critical situations.

### 2.3 Monitoring Agents

Monitoring agents are responsible for assisting the commander in maintaining an accurate awareness of the current situation (situational awareness) at all times. The amount of information available to battlefield commanders will continue to increase to the point of informational overload. The main role of monitoring agents will be to prevent information overload by fusing, filtering, and prioritizing raw data, and transforming that data into information that the commander can use in the context of the current situation. For example, different units may report directional vectors for the source of sniper fire. The monitoring agent could use this vector data to triangulate the sniper's position and rec-

ommend through the tasking agent that indirect suppressing fire be called on that location. Another possible data fusion role could be more proactive. Monitoring agents could use templates such as intelligence formats (e.g., SALUTE reports, which specify the Size, Activity, Location, Unit, Time, and Equipment of an observed enemy) to task sensors or prompt humans for missing fields.

Monitoring agents should also filter information, especially when the commander is engaged in critical tasks, to minimize distractions. For example, if the commander is busy responding to an ambush with one unit, he probably doesn't care at the time that another unit's status is "Okay" and has not changed. Such routine status reports should be stored for future reference, but kept in the background so as to not interfere with more important tasks. Likewise, such information can be prioritized by criticality or by relevance to current commander tasks. For instance, message traffic and information flow may increase dramatically during a firefight. Where loss of life or equipment is imminent, relevant information that might prevent or mitigate the situation could me made more salient for the commander (e.g., by color or ordering in a message list). Other monitoring agent tasks might include:

- Automatically updating and synchronizing COP (common operational picture) databases.
- Presenting appropriate data visually, such as unit location, direction, supply levels, and damage status.
- Providing all messages relating to a single friendly or enemy unit to help build a broader picture from single events.
- Represent visually direct communication lines between shooters and sensors.
- Monitoring health and stress levels of human subordinates.

## 3. System Environment

The current CIANC$^3$ system integrates Soar-based interface agents into a combined simulation and operational environment for robotic control. The agents communicate using the FIPA protocol and a user interface to the agents was created using Tcl.

### 3.1 Integration Environment: Operator Control Unit / OneSAF TestBed

Bialczak, Nida, et al. (SESI, 2000) have designed and implemented a dynamic composable Operator Control Unit (OCU) for the Mounted Maneuver Battle Lab (MMBL) (now Unit of Action Maneuver Battle Lab, UAMBL) at Fort Knox Kentucky. Requirements for the OCU included that control of the unmanned vehicles had to be dynamic (i.e., had to transition from one OCU to another without interrupting the exercise), and that the OCU had to be composable – it had to control a heterogeneous set of robots. Most of all the OCU had to be easy to use. Because of its inherent flexibility, they were also able to task several real robotic vehicles from the OCU. The OCU mixes the simulation functionality of OTB and the control capability of a robotic control system. This combination makes the OCU uniquely qualified as a platform on which to further develop the CIANC$^3$ system.

### 3.2 Agent Environment: The Soar Cognitive Architecture

The Soar cognitive architecture is a powerful framework for creating multi-agent systems. Soar has been successfully used to model complex battlefield engagements in field simulations. Soar was used to create synthetic agents for FWA, RWA, and related controllers. For example, we have created Soar models of fighters and strikers that interact with Soar forward air controllers during close-air support simulations. Similarly, for defensive-counter air (DCA) missions, Soar-based fighters coordinate with a Soar-based Airborne Early Warning (AEW) agent (currently in a simulated E-2C) that provides broadcast and close control support to fighters. In all cases, human operators can also provide command and control to Soar agents. This intervention is allowed but not required. Recent work has developed a model of ground forces (Taylor, Koss, & Nielsen, 2001).

### 3.3 Agent Communications: FIPA

Robotic forces must be able to communicate with each other in order to conduct joint operations. An agent communication language (ACL) provides a common way for agents to communicate. An effective ACL must enable interface agents to communicate between multiple echelon hierarchies of both robotic and human forces. The Foundation for Intelligent Physical Agents (FIPA) (Huhns, 1997) has defined an agent communication lan-

guage that will enable robotic forces to perform these types of communication. The FIPA standard also offers several additional benefits. The FIPA ACL provides a formal semantics that allows interface agents to deal with actions explicitly. This will enable robotic forces to make decisions, maintain situation awareness, and share information more efficiently. By using a FIPA-based ACL, robotic forces will be able to execute commands rapidly, and describe their actions precisely. Robotic forces will be able to share awareness information about their current situation, status, plans, and experiences. This will allow groups of robotic forces to coordinate activity. The FIPA ACL also provides explicit support for secure communication, making it more difficult for enemy forces to compromise robotic force communications.

## 4. Conclusions

The agent and communication system designs were successfully implemented in a simulation environment. A scenario was created to test the system using a simple combination of a sensor-vehicle (UAV) and a shooter-vehicle (UGV). The UGV took the tasking to seek and destroy a suspected enemy. The UGV tasked a UAV to locate and acquire the target. The UAV located the target and transmitted the coordinates to the UGV, which then confirmed with the human operator before firing on, and destroying the target.

The scenario was simple enough to test and demonstrate the capabilities of the interface-agent architecture, but it was not complex enough to demonstrate any real utility to robotic controllers. In addition, the Tasking agent accomplished most of the background work. A more complex scenario would place more demands on other agents, driving their further elaboration.

The UML sequence diagram (see figure 2) demonstrates how complex the agent communication protocols need to be in order to perform even the simple scenario. Implementing a more complex scenario will determine whether the sequence diagram is enough to simplify that portion of the development or whether additional tools will be required.

The performance of the Soar cognitive architecture was more than adequate for the agent task in the simple scenario. However, implementing the agent behaviors was somewhat complex, even with Soar. More research effort needs to be spent (outside of this project) developing tools and techniques for rapid agent development. Research conducted in the next phase of this project should help to better define the requirements for such tools. Related to this is the large amount of declarative knowledge that needs to be encoded into the Soar agent procedures. Representing such knowledge within production rules will inhibit scaling because changes to that knowledge will necessitate time-consuming and expensive work. We should explore an ontological approach to representing such knowledge to disentangle the behavioral knowledge from the declarative knowledge.

Although FIPA provided a great foundation for developing the agent communication infrastructure, it is not clear that it will meet the needs of military systems. For Phase II we will consider using DARPA's CoABS grid for the agent communication architecture.

In summary, Phase I successfully demonstrated the technical feasibility of interface agents for robotic command and control. It also provided the necessary infrastructure and techniques necessary to rapidly explore much more of the problem space.

## 5. Future Work

In Phase II of the CIANC$^3$ project we will continue the agent architecture and communications research begun in Phase I, focusing on the development of flexible interaction models that support rich analysis processes and human interactions with the goal of developing a novel, tightly integrated semantic approach to communicative acts and conversation protocols. In addition we will develop an improved GUI interface to better facilitate communication and interaction between human and software agents.

To determine the efficacy of our approach, it is imperative that the system be tested. These tests will take the form of small-scale individual usability tests at UAMBL or ARI, actual use of the prototype at UAMBL, or a full-scale comparison of company-level training pitting a CIANC$^3$ enhanced FCS-company against a conventional company. In any case, this phase of the project will be dedicated to the testing design & performance testing of the overall system.

## 6. Acknowledgement

This project has been supported by the Army Research Institute, contract number DASW01-02-C-

0019. We would like to thank Dr. Carl W. Lickteig, COTR, Scott Shadrick and the other staff at the ARI Armored Forces Research Unit.

## 7. References

[1] DARPA. (2001). *Future Combat Systems Solicitatio*n. DARPA. Available: www.darpa.mil/fcs/Solicit.html [2001, July].

[2] Horvitz, B. *Principles of Mixed-Initiative User Interfaces*. In Proceedings of CHI'99, ACM SIGCHI Conference on Human Factors in Computing Systems, May 1999.

[3] Laurel, B. (1991). Interface Agents: Metaphors with Character. In B. Laurel (Ed.) *The Art of Human-Computer Interface Design*, (pp. 347-354). Reading, MA: Addison-Wesley Publishing Company, Inc.

[4] Lieberman, H. (1997). Autonomous Interface Agents. In *Proceedings of the ACM Conference on Computers and Human Interaction [CHI-97]*, Atlanta, March 1997: ACM Press.

[5] Science and Engineering Services, Inc. (2000). Multi-Functional Operator Control Unit, (U.S. Army Armor Center and School Contract DABT-23-00-D-1035) Radcliff, KY.

[6] Taylor, G., Koss, F., Nielsen, P. (2001). Special operation forces IFORs. Proceedings of the Tenth Conference on Computer Generated Forces. May, 2001.

[7] Huhns, M. N., & Singh, M. P. (1997). Agents and Multiagent Systems: Themes, Approaches, and Challenges, *Readings in Agents* (pp. 1-23). San Francisco, CA: Morgan Kaufmann Publishers, Inc.

## Author Biographies

**DR. SCOTT WOOD:** Dr. Wood is a Senior Scientist with Soar Technology with over ten years of research and industry experience in the areas of software development, e-business consulting, cognitive modeling, and human-computer interaction. Dr. Wood also has extensive experience developing human-performance models using the EPIC cognitive architecture, optimizing workflows and interface usability through task analysis, and in designing web solutions for e-business applications. He also served four years in the U.S. Army. He earned a B.S. in Computer Science (1990) from Tulane University, and M.S. (1994) and Ph.D. (2000) degrees in Computer Science and Engineering from the University of Michigan, Ann Arbor.

**JACK ZAIENTZ:** Jack Zaientz is a Research Scientist for Soar Technology focusing on human computer interaction issues involving interface design, information visualization and agent systems. He has over seven years experience in human computer interaction, analyzing, designing and implementing applications and application interfaces for a wide range of industries and has earned a Masters of Human Computer Interaction from Carnegie Mellon University's Human Computer Interaction Institute.

**JONATHON BEARD:** Jonathan T. Beard is a Software Engineer and the TacAir-Soar Product Manager at Soar Technology. He is currently involved in behavior-model and systems-interface research & development. He has over ten years of experience in the computer industry, including software development, systems and network administration, database design and administration, and professional consulting. His research interests are in artificial intelligence, wearable and embedded computing, linguistics, and cooperative autonomous agents. He has attended Hope College, the Kyrgyz State National University, and Northern Michigan University.

**DR. RICHARD FREDERIKSEN:** Dr. Frederiksen, obtained his PhD in Aerospace Engineering from the University of Michigan in 1996. Prior to joining Soar Technology, he was the lead technical programmer for NGB Technologies, where his duties included developing advanced simulation tools for various aerospace applications, including combustion and fluid dynamics simulations. His work at Soar Technology has centered on the application of game technology to various advanced simulation, and development of new agent management tools. He earned MSE (1993) and BSE (1990) degrees in Aerospace Engineering at the University of Michigan.

**DR. MARCUS HUBER:** Dr. Huber has a strong academic background in artificial intelligence covering intelligent agent architectures, multi-agent coordination, cooperative distributed problem solving, plan recognition, reasoning and decision-making under uncertainty, and agent communication language semantics and implementation. He received a B.S. in Electrical Engineering (1988) from GMI Engineering and Management Institute, and an M.S. (1991) and Ph.D. (1996) degrees in Computer Science from the University of Michigan.