Technical Report 1134

Cooperative Interface Agents for Networked Command, Control, and Communications (CIANC³): Phase I Final Report

Scott D. Wood
Soar Technology, Inc.

April 2003



United States Army Research Institute for the Behavioral and Social Sciences

Approved for public release; distribution is unlimited

AUTHENTICATED
U.S. GOVERNMENT
INFORMATION
GPO

U.S. Army Research Institute for the Behavioral and Social Sciences

A Directorate of the U.S. Total Army Personnel Command

ZITA M. SIMUTIS
Director

Research accomplished under contract
for the Department of the Army

Soar Technology, Inc.

Technical Review by

Robert J. Pleban, ARI
Paul A. Durlach, ARI

NOTICES

DISTRIBUTION: Primary distribution of this Technical Report has been made by ARI. Please address correspondence concerning distribution of reports to: U.S. Army Research Institute for the Behavioral and Social Sciences, Attn: TAPC-ARI-PO, 5001 Eisenhower Ave., Alexandria, VA 22333-5600.

FINAL DISPOSITION: This Technical Report may be destroyed when it is no longer needed. Please do not return it to the U.S. Army Research Institute for the Behavioral and Social Sciences.

NOTE: The findings in this Technical Report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
April 2003	Final	3/1/02 – 8/31/02

4. TITLE AND SUBTITLE

Cooperative Interface Agents for Networked Command, Control, and Communications (CIANC^3): Phase I Final Report

5a. CONTRACT NUMBER
DASW01-02-C-0019

5b. GRANT NUMBER
0602785A

5c. PROGRAM ELEMENT NUMBER
A790

6. AUTHOR(S)
Scott D. Wood (Soar Technology, Inc.)

5d. PROJECT NUMBER
211

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Soar Technology, Inc
3600 Green Court, Suite 600
Ann Arbor, MI 48334

8. PERFORMING ORGANIZATION REPORT NUMBER 20020301

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Institute for the Behavioral and Social Sciences
ATTN: TAPC-ARI-IK
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

10. SPONSOR/MONITOR'S ACRONYM(S)
ARI

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Technical Report 1134

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES
Contracting Officer's Representative: Carl W. Lickteig

14. ABSTRACT
Report developed under a Small Business Innovation Research Program 2000.2 contract for topic A02-024. The research reported here explored methods for effectively controlling FCS units containing mixed human and robotic elements. The objective was to determine whether an agent framework built around three specified agent types (Tasking, Coordinating, and Monitoring) could be constructed to add an intelligent abstraction layer between human commanders and battlefield elements. The focus was to identify human-system interaction issues, design potential solutions, and create software that supports the commander's tasks and mitigates inherent human performance limitations. A prototype interface agent architecture was designed, and a framework was implemented. Interface agents were created to perform in a simple, simulated battle scenario. The work conducted during Phase I lays the foundation for a Phase II plan to create more realistic scenarios and test the utility of interface agents in a variety of experimental settings.

15. SUBJECT TERMS
Training, Interface Agents, Intelligent Agents, Command and Control, Future Combat Systems

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Carl W. Lickteig
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified	Unlimited		19b. TELEPHONE NUMBER *(include area code)* 502–624-2613

Technical Report 1134

COOPERATIVE INTERFACE AGENTS FOR NETWORKED COMMAND CONTROL, AND COMMUNICATIONS (CIANC^3): PHASE I FINAL REPORT

Scott D. Wood
Soar Technology, Inc.

Armored Forces Research Unit
Barbara A. Black, Chief

U.S. Army Research Institute for the Behavioral and Social Sciences
5001 Eisenhower Avenue, Alexandria, Virginia 22333-5600

February 2003

Army Project Number
2O262785A790

Personnel Performance and
Training Technology

Approved for public release; distribution unlimited.

FOREWORD

As Army weapons systems become increasingly complex, we must ensure that training can keep pace. This is a primary mission of the U.S. Army Research Institute for the Behavioral and Social Sciences (ARI). Are there ways of improving trainability by mitigating inherent complexities? One promising technique for doing this is through highly-usable intelligent user interfaces.

This Phase I Small Business Innovation Research Program (SBIR) effort explored the feasibility of interface agents for simplifying the command and control of multiple unmanned vehicles. The research goal was to design a suitable architecture for exploring and creating interface agents compatible with the Army's Command, Control, Communication, Computer, Intelligence, Surveillance, and Reconnaissance (C^4ISR) systems. This architecture was implemented and tested in simulation using a simple battle scenario. This initial test of technical feasibility allows for further research to determine how such interface agents could be used, whether they can truly reduce operator workload and improve performance, and the parameters of such usage within the context of robotic command and control. Determining the factors relevant to intelligent task support is critical for the Future Combat Systems program if it is to deliver on its promise of reducing manpower and dramatically improving the Army's force capabilities.

This research was part of ARI's Future Battlefield Conditions (FBC) team efforts to enhance soldier preparedness through development of training and evaluation methods to meet future battlefield conditions. This report represents efforts for Work Package 211, Techniques and Tools for C^4ISR Training of Commanders and Staffs in Future Combat Systems Units (FUTURETRAIN). Results of this effort and plans for the Phase II effort were reviewed by representatives from the Army's SBIR program. As a result of the Phase I success, the Phase II effort was awarded, and an interface agent system should be available for commercial application by January 2005.

KATHLEEN A. QUINKERT
Acting Technical Director

ACKNOWLEDGEMENTS

This report describes work performed under a Small Business Innovation Research Program 2000.2 contract for topic A02-024. The project goal was to explore methods for effectively controlling FCS units containing mixed human and robotic elements with a focus on training implications. The success of this project reflects the efforts of many individuals.

I would like to thank the team at U.S. Army Research Institute for the Behavioral and Social Sciences (ARI) Fort Knox Field Unit. In particular, I would like to thank Dr. Carl Lickteig for patiently guiding us through the process, sharing his knowledge and expertise, and helping to make useful connections to other collaborators. I would also like to thank Mr. Scott Shadrick for using his unique blend of psychological and technical skills to focus our efforts in both areas.

I would also like to thank Dr. Robert Bialczak and Mr. Jon Nida of Technology Services Associates for their considerable support with their Operator Control Unit (OCU) for robotic control. Their efforts allowed us to start with a more realistic interface environment; one that is used to control both actual and simulated robotic entities. I also thank Dr. Russell Vane of Veridian Systems for his domain expertise in Armored Cavalry operations.

Finally, I would like to thank the Soar Technology CIANC3 team, Mr. Jonathon Beard, Dr. Richard Frederiksen, and Mr. Jack Zaientz, for their scientific expertise, creative abilities, and professional standards. Dr. Marcus Huber of Intelligent Reasoning Systems was instrumental in developing a communication architecture and clearly defining a workable scenario. I would also like to thank the others at Soar Technology for their considerable support.

COOPERATIVE INTERFACE AGENTS FOR NETWORKED COMMAND, CONTROL, AND COMMUNICATIONS (CIANC^3): PHASE I FINAL REPORT

EXECUTIVE SUMMARY

Research Requirement:

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 include 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 led 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 has been to identify the human-interface issues, design potential solutions and create software that supports the commander's tasks and mitigates human performance limitations in the context of robotic command and control.

The overall research goals for this project were to determine whether a specific class of software agent, autonomous intelligent interface agents, could be created to reduce the complexities inherent in controlling multiple unmanned vehicles. Furthermore, if such a system could be built, how could it best be used and would it fundamentally improve operator performance. Finally, if such a system were useful, how could future warfighters be effectively trained to control multiple robots.

The technical objective for Phase I of this project was to demonstrate the feasibility of a cooperative multi-agent system for control of battlefield robots. That is, the project was to determine whether an agent framework built around the three specified agent types, tasking, coordinating, and monitoring, could be constructed to add an intelligent abstraction layer between human military commanders and robotic battlefield entities.

Procedure:

Under Phase I of this Small Business Innovative Research contract, we researched methods for effectively controlling FCS units containing mixed human and robotic elements. Our objective was to determine whether an agent framework built around three specified agent types (Tasking, Coordinating, and Monitoring) could be constructed to add an intelligent abstraction layer between human commanders and battlefield elements. The focus was to

identify human-system interaction issues, design potential solutions, and create software that supports the commander's tasks and mitigates inherent human performance limitations.

During the initial phase of this project we progressed in several areas. Major accomplishments of Phase I included:

- A working scenario was defined on which to determine project feasibility.
- Architectural tradeoffs were discussed culminating in a working system and communication architecture.
- Necessary inter-agent communications protocols to perform the scenario were designed.
- Soar was integrated into the Operator Control Unit (OCU)/OneSAF TestBed (OTB) code base.
- A Soar-language prototype multi-agent C^3I support system was developed and tested.
- A prototype user interface for the multi-agent system was designed and implemented.

Findings:

The agent and communication system designs were successfully implemented in a simulation environment, the Operator Control Unit (OCU). 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 will place more demands on the Coordinating and Monitoring agents, driving their further elaboration.

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.

Utilization of Findings:

Phase II will expand the target scenario by implementing some portion of an Army approved FCS scenario. Phase II will further develop the agent and communication architectures, the agents, domain-specific behaviors, and the user interface. Phase II will integrate speech and voice recognition into the OCU and instrument it to collect human and system data. Phase II will identify and explore a variety of human factors involved in robotic C^2 and conduct task analyses as necessary. At appropriate points in the project, the system will be evaluated by civilian and military personnel. By the end of Phase II, January 2005, the Army and commercial sectors should benefit from an interface architecture that helps manage robotic systems and reduce users' knowledge and training requirements.

COOPERATIVE INTERFACE AGENTS FOR NETWORKED COMMAND, CONTROL, AND COMMUNICATIONS (CIANC^3): PHASE I FINAL REPORT

CONTENTS

	Page
PHASE 1	1
Identification and Significance of the Problem	1
Background	2
Robotic Battlefield Entities	2
Human-Machine Interaction and Supervisory Control	3
Agents and Multi-Agent Systems	3
Interface Agents	6
Phase I Technical Objectives and Approach	6
The Vision: Cooperative Teams of Soar Interface Agents	6
Technical Objectives	7
Approach	7
CIANC^3 Agent Roles and Responsibilities	9
System Implications	11
Usability and Training Implications	12
Phase I System Environment	13
Integration Environment: Operator Control Unit / OneSAF TestBed	13
Agent Environment: The Soar Cognitive Architecture	14
Agent Communications: FIPA	15
User Interface Development: Tcl	16
Phase I Activities and Accomplishments	16
Conceptual FCS Vignette	17
Phase I Scenario	18
System and Communication Architecture	19
Inter-Agent Communications for the Scenario	22
Soar-OCU/OTB Integration	23
Soar Prototype Agent	23
User Interface Prototype	23
CIANC Agent Behavior in the NLOS Bombardment Mission	25
Summary of Phase I Results	27

CONTENTS (Continued)

Page

PHASE II 28

Phase II System Design 28

Infrastructure Design 28
Design Concept 28
The Messaging Infrastructure 29
Multi-Agent Design 31

Phase II Work Plan 32

Research HSI, Scenarios, Behaviors 33
Develop GUI 33
Design and Conduct System and Human Tests 34
Develop Testing Environment 34
Research Agent Architecture and Communications 35
Develop Agent Architecture and Communications 35
System Development and Integration 35
Develop Agent Behaviors 35

PHASE III TRANSITIONAL PLAN 35

Phase III Commercialization Strategy 35

REFERENCES 39

APPENDIX A: Agent Message Types A-1

APPENDIX B: Agent Communications UML Sequence Diagram B-1

LIST OF TABLES

Table 1. Scenario phases and entity responsibilities 18

LIST OF FIGURES

Figure 1. OTB movement dialog box 2
Figure 2. CIANC^3 Conceptual Overview 8
Figure 3. Phase I Scenario Narrative Map 17
Figure 4. Phase I Design Option I 20
Figure 5. Phase I Design Option II 20
Figure 6. Phase I Design Option III 21

Page

Figure 7. Phase I Design Option IV 22
Figure 8. Inter-agent communication diagram 23
Figure 9. The Task Definition Pane of the user interface prototype 24
Figure 10. The Task Monitoring Pane of the user interface prototype 25
Figure 11. Messaging sequence diagram 30
Figure 12. Agent registration and status messaging 31
Figure 13. Experimental Environment 34

COOPERATIVE INTERFACE AGENTS FOR NETWORKED COMMAND, CONTROL, AND COMMUNICATIONS (CIANC^3): PHASE I FINAL REPORT

Phase I

Identification and Significance of the Problem

The vision of the future for armored and mechanized military structure, as spelled out by (Defense Advanced Research Projects Agency, 2001), includes the use of mixed teams of human and robotic forces on a dynamic and rapidly changing battlefield. Implementing the vision will include a shift from complete human control of weapons systems to 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. Two keys to this will be the degree to which teams and individual robots are autonomous, and whether the commander's human-machine interface is designed such that the commander has superior awareness of the situation. Heightened awareness will afford the commander the ability to rapidly create and execute battle plans. The focus of this project has been to identify the human-interface issues, design potential solutions and create software that supports the commander's tasks and mitigates human performance limitations.

To illustrate the limitations of current control technologies, consider how OneSAF Test bed (OTB) users currently specify the behavior of simulated semi-autonomous entities. Figure 1 shows a typical OTB screen for specifying the travel orders for a simulated battlefield entity. There are 15 main sections of information for input. Even if the user were able to complete each section in 4 seconds it would require at least 1 minute to complete this screen. Firing orders and other necessary plans and contingencies would take additional time. While this level of performance may be acceptable for simulation purposes, it is clearly not acceptable for battlefield performance, especially considering that such steps might be necessary for each vehicle. The main point is that time to respond to various battlefield events (such as a movement order) is likely to be much longer than time available. If instead, the commander had given the order to a human staff assistant, the assistant would be able to determine many of the details necessary to specify the commander's intent. For example, unit type, readiness state, enemy proximity and disposition, and mission type could be used to infer much of the information required by the OTB dialog box. Since much of the information required can be determined from unit standard operating procedure (SOP) or can be inferred by a subordinate from the mission profile, this type of command and control problem lends itself to expert system solutions and/or other forms of performance enhancing technologies.

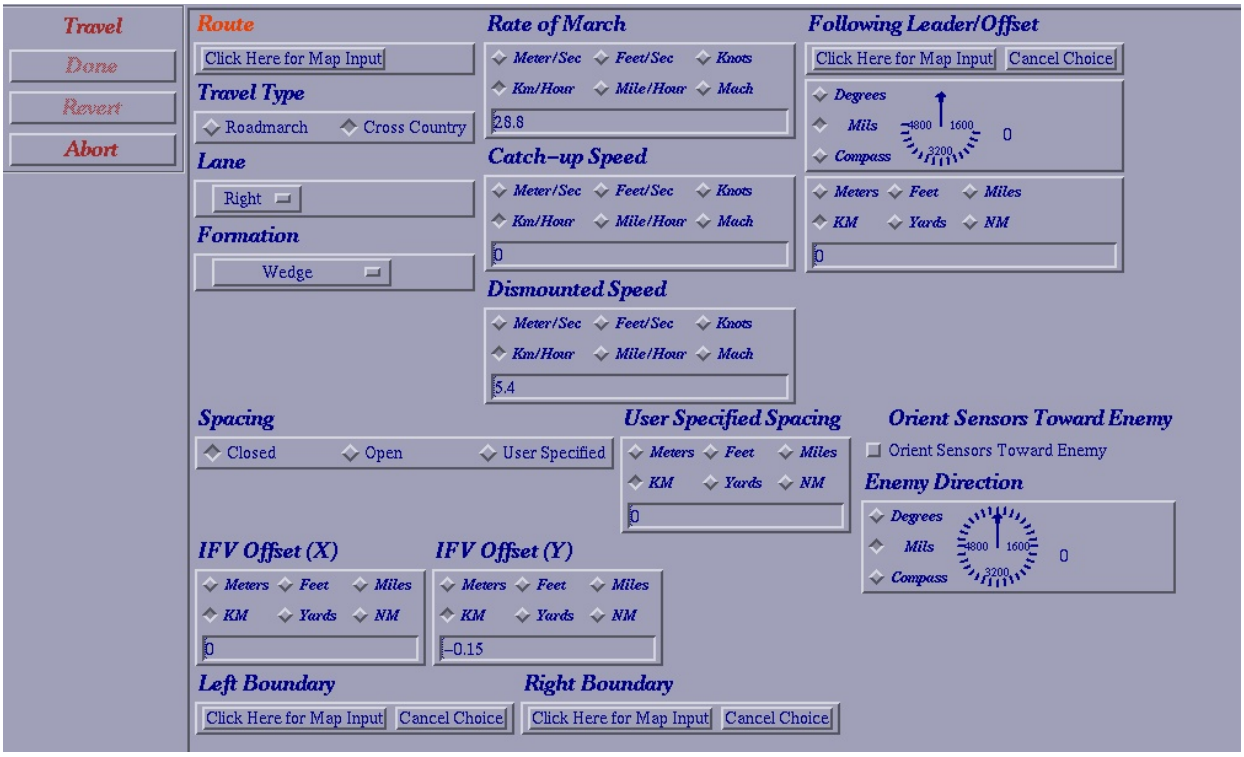


Figure 1. OTB movement dialog box. To specify travel orders for a simulated entity, the user completes the fields. With 15 main sections, completing all the necessary data is too slow for battlefield performance.

Military commanders will face similar command and control issues when leading mixed units of live and robotic entities. The problem is that current unmanned systems require too much human control to meet the goal of one controller per multiple battlefield entities. For example, the Predator unmanned aerial vehicle requires a minimum of three people to operate it, one officer and two senior non-commissioned officers. In order to meet the force transformation goals, the information and interaction needs of future robotic systems will need to be reduced to allow multiple robots to be controlled by a single human. This goal can only be realized through increased robotic autonomy and improved human-machine interaction. We propose to build on our Phase I efforts to further develop an innovative system of cooperative intelligent interface agents for network command, control, and communication.

Background

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, 2001; 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, 2001). 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.

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. Two approaches that have been taken to improve usability are direct-manipulation and intelligent interfaces. Direct manipulation interfaces stress the accurate visualization of large amounts of data (Shneiderman & Maes, 1997). These interfaces also enable users to directly manipulate and select data by pointing, and other rapid, incremental, and reversible actions that provide immediate feedback. 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.

Agents and Multi-Agent Systems

The challenge of developing complex agent-based systems is to first determine the most appropriate representation and reasoning framework for individual agents and then to determine how the many agents will communicate and cooperate. Individual agents can vary greatly in the expressiveness and power of their knowledge, procedural representations, reasoning algorithms, and capabilities. Multi-agent systems can vary widely in the manner in which the agents are organized, the language and protocols used to interact, and the coordination and problem solving paradigms they utilize to complete their tasks. This section will clarify what is meant by an "intelligent agent," discuss agent frameworks, and introduce a number of issues inherent in all multi-agent systems.

What do we mean by the term "intelligent agent?" An informal, intuitive definition is: "intelligent" - having knowledge/expertise, and "agent" - a software-based process that can do a task for someone or something else (e.g., a travel agent).

The following is a more formal, yet still quite weak, definition of agents that is adapted from (Wooldridge, 2000):

Agent - a software-based computer system that has the following properties:

- Autonomy - acts without the direct intervention of humans or others and with some control over their actions and internal state.
- Social ability - interacts with other agents and possibly humans.
- Reactivity - responds in a timely manner to changes in the environment.
- Pro-activeness - exhibits goal-directed behavior.

A stronger notion of agency is based on the Beliefs-Desires-Intents (BDI) framework (Wooldridge, 2000). It is widely held by the Artificial Intelligence (AI) community that, in addition to the weak, formal properties, the software process must conceptually or explicitly embody humanistic characteristics such as:

- Knowledge and Beliefs - facts or belief about environment (including other agents) or internal state.
- Desires and Goals - motivations for acting.
- Intentions - commitments to courses of action based upon motivations.
- Obligations - commitments to other agents.
- Rationality - actions are purposeful toward achievement of goals.

The agent research and development for this effort, and the specific expertise of Soar Technology and Intelligent Reasoning Systems, focuses primarily on building agents that embody this stronger notion of agency.

There are many "agent" architectures such as Aglets and JAVA Agent Template (JAT) from IBM, Agent Tcl (Gray, Kotz, Cybenko & Rus, 1997) from Dartmouth, and Agents for Remote Action (ARA) (Peine & Stolpmann, 1997) from the University of Kaiserslautern that have little or no explicit representations of any of the mentalistic attributes of beliefs, intentions, capabilities, or even goals (often argued to be the key feature of agency). Furthermore, programmers encode the behavior of these agents almost completely through low-level hard coding. Each of these agents provides specialized functionality in some focus area (e.g., ARA and Agent Tcl are specialized to provide mobility capabilities) but do not otherwise provide what we consider a complete reasoning architecture.

Agent architectures such as Soar (Laird, Newell & Rosenbloom, 1987) and AOP-based (Agent-Oriented Programming) (Shoham, 1993) architectures such as Agent-0 (Shoham, 1991), LALO, and PLACA (Thomas, 1995) all provide significantly more complete representation and reasoning frameworks than those mentioned in the preceding paragraph. Soar implements a

unified theory of cognition (Newell, 1990) and provides a wide range of desired agent architecture capabilities, including integrated execution, means-ends planning, meta-level reasoning, and learning. The AOP-based architectures provide explicit internal representations of mentalistic concepts such as beliefs and commitments but they emphasize social interaction capabilities over individual capabilities (even though PLACA does extend the AOP paradigm to include generative planning capabilities).

In addition to these primarily monolithic agent architectures are a number of multi-level agent architectures such as Touring Machines (Ferguson, 1992), Atlantis (Gat, 1992), and InteRRap (Muller & Pischel, 1994). Agent architectures of this style vary widely in their theoretical foundation, internal representations, architectural components, and particular emphasis on specific representational or behavioral issues or application domain. One common problem with multi-layer architectures is that they require a specialized programming language for each layer (e.g., reactive layer language, scheduling layer language, planning layer language, coordination layer language). None of these architectures provides as mature or cohesive a theoretical basis as that provided by the BDI theory.

There are many challenging issues that must be addressed when developing multi-agent systems. This includes how the agents are organized and what role the agents play within the organization (Birmingham, D'Ambrosio, Darr & Durfee, 1994; Fox, 1988). Within the FCS system, much of the agents' organization will be dictated by military doctrine. However, with multiple agents associated with each unmanned vehicle operator and the possibility of combat losses, the static and dynamic organization and role determination (Corkill, 1982; So & Durfee, 1994; So & Durfee, 1997) will be important issues to address.

Another important issue in multi-agent systems is determining what communication language semantics and syntax the agents will use at both the performative and content level (FIPA: Foundation for Intelligent Physical Agents, 2000; Labrou, 1996; Cohen & Levesque, 1990; Huber, 1999). The performative level is associated with the intention of the message, such as whether it is a directive (command, question, or request), an assertive (information/knowledge passing), a commissive (commitment forming), etc. (Searle, 1970). The content level is associated with the specifics of the communication, such as the task being requested or the information being passed, and is almost always domain specific.

Entities within organizations tend to interact with each other in regular, standard patterns and this holds true for intelligent agents as well. These interaction patterns simplify agent reasoning by constraining agent behavior and facilitate creation of expectations and standard behavior models of other agents. Capturing these patterns, commonly called conversation policies or interaction protocols (Bradshaw, Dutfield, Benoit & Wooley, 1997; FIPA, 2000; Kumar, Cohen & McGee, 2001; Labrou & Finin, 1997), is required in any complex multi-agent environment and needs to reflect, for example, any authority relationships that exist between agents (John & Kieras, 1996).

The manner in which the agents work together to complete their tasks is crucial to the agents' performance in any domain, and has been the topic of a great deal of research. There are many factors involved with determining the problem-solving paradigm of the multi-agent

system. Just a few issues include whether problem solving is done in a centralized or decentralized manner (Fox, 1988; Durfee, Kenny & Kluge, 1998), whether tasks are distributed or can be handled by a single agent (Gasser & Hill, 1990), the level of robustness and fault tolerance required in the domain (Kumar & Cohen, 2000; Rosenschein, 1985), the level of uncertainty and rate of change in the environment (Fox, 1979), whether a static problem solving scheme will be used or whether the problem solving scheme can be dynamically changed (Decker & Lesser, 1995; Rosenschein, 1985).

Interface Agents

Interface agents (Laurel, 1991) are a specific form of agents 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. While some agents operate solely in the background, interface agents are designed as user interface elements that can directly assist users with their tasks. This can include assistance with input tasks that facilitate the specification of complex commands to decrease task execution time and improve accuracy. Interface agents can also assist with information output, interpreting raw data or filtering necessary information from non-relevant data.

A weakness of some of the previous work on intelligent interface agents is that human operators needed a significant amount of training and they had to think in terms dictated by the software agents. A goal of intelligent interface design is to make the interface invisible (Maes, 1994). This is not to say that interface elements can ever fully disappear, but rather that the translation between the user's mental model of the task and the computer's model for the task is minimized. This idea is a core tenet of usability: The user should be able to focus on the primary task, and not the technology used to accomplish that task. One way of approaching this is by merging software-agent technology with proven direct manipulation techniques (Shneiderman & Maes, 1997).

Phase I Technical Objectives and Approach

The Vision: Cooperative Teams of Soar Interface Agents

Interface agents were constructed using production rules in the Soar Cognitive Architecture. An agent communications protocol was developed using FIPA as the language basis. In general, agents were designed such that declarative information, such as weapon characteristics, are stored in data files so that it can easily be modified. Procedural knowledge, such as tactical heuristics, was encoded using productions. The procedural knowledge was modularized within Soar to allow for maximal reuse of all encoded knowledge.

Each of the agent types was also modularized according to the roles they play. Tasking agents are designed to incorporate reasoning about order types, weapon types, enemy weapon types, and other necessary information. Coordinating agents include knowledge of

communication formats and procedures, and heuristics for coordinating with other units. Monitoring agents include knowledge of event types, how to prioritize them, how to filter them, etc. In addition, there is also the need for a meta-layer of agent knowledge that specifies rules for inter-agent communication.

Technical Objectives

The technical objectives for this SBIR are to demonstrate the feasibility of a CIANC^3-like system for control of battlefield robots. That is, the project will determine whether an agent framework built around the three specified agent types can be constructed to add an intelligent abstraction layer between human military commanders and robotic battlefield entities. Phase I demonstrated feasibility on a technical level. Phase II will test whether such a system will actually benefit FCS commanders. There are many issues that will not be specifically addressed, such as, what are the best interaction techniques for optimal object selection, or how best can this system be integrated into the military command structure. The technical objectives of this project are:

1. Determine human information needs for controlling mixed human and robotic teams.
2. Determine appropriate levels of automation for human tasks that will reduce cognitive workload yet maintain sufficient human control.
3. Determine suitable high-level architecture for interface agent organization and develop inter-agent interaction protocol.
4. Develop usable human interface to software agents that will demonstrate agent interactions, demonstrate abstract-to-concrete command translation, and allow testing of target scenario.
5. Determine scalability of system and develop more complex scenario to demonstrate these capabilities.
6. Further demonstrate the feasibility of the concept and explore real-world issues by integrating the prototype into a robotic control and simulation system.
7. Test the system for usability, performance, and using a variety of engineering and psychological methods.

Approach

The approach we have taken to achieve the technical objectives is to create a framework of cooperative interface agents for networked command, control, and communication. The initial goal of performance-enhancing technologies for command and control should be to start with the current roles found in current command staffs to augment such activities and eventually move further, providing real-time situation awareness and decision support beyond what is humanly possible. Command staffs commonly provide five basic functions to commanders in support of reconnaissance, security, offensive, and defensive operations (c.f., 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.

This project demonstrates the feasibility of creating a suite of interface agents that can provide functionality currently provided by command staffs. The Command and Control Hierarchy, illustrated in Figure 2, shows how intelligent agents can provide a layer of abstraction between command echelons as well as between human controllers and robotic control systems. Special-purpose agents are encapsulated into a meta-level agent to facilitate internal agent communication and information sharing. The interface agents control the flow and form of much of the information to be displayed to the human operator in a mixed-initiative dialog. The human operator uses direct manipulation techniques to interact with the agents. If successful, the human operator will not be aware of the underlying agent technology or the separation of agent roles.

The Command and Control Hierarchy functions are divided between three classes of agents: tasking, monitoring, and coordinating. 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. Regardless 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.

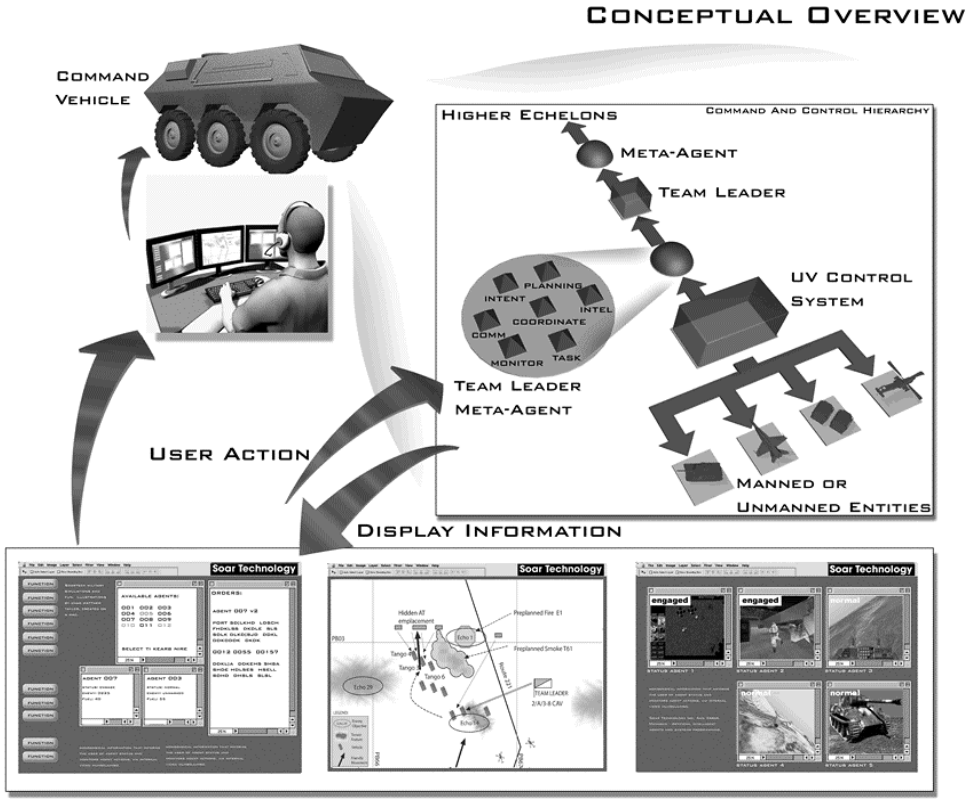


Figure 2. CIANC[3] Conceptual Overview.

The FCS vision of future armored and mechanized military structure includes use of mixed teams of human and robotic forces on a dynamic, rapidly changing battlefield. This will require an overall force reduction with multiple entities 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 both human and robotic-teams to be controlled quickly and easily. Key to this will be the degree to which teams and individual robots are autonomous, and whether the commander's human-machine interface is designed so the commander has superior control and awareness of the situation. The initial phase of this effort addressed the issue of whether an agent framework built around the three specified agent types (Tasking, Coordinating, Monitoring) could be constructed to add an intelligent abstraction layer between human military commanders and robotic battlefield entities. The focus was to identify the human-interface issues, design potential solutions and create software that supports the commander's tasks and mitigates human performance limitations.

CIANC^3 Agent Roles and Responsibilities

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.

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, identify friend foe (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.

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 recommend 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 be 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.

System Implications

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. It should also be assumed that the target agent organization described here will 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 commanders current task from recent actions.
- Deriving enemy intent based on recent enemy actions.
- Red-teaming plans.

- Routine scheduling of communications, supply, and duty rotations.

In addition, 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 should be included in later phases of the project.

Usability and Training Implications

To determine the proper tasks to automate, the necessary information requirements, and how to optimize human procedures, designers must gain a deep understanding of the user's task. Task analysis (Kirwan & Ainsworth, 1992) is used to capture, understand, and improve existing human procedures, and how human tasks can best be improved with technology. In addition, a task analysis will spell out a system's training requirements, specify the steps and ordering of training materials, and provide the basis for task-based help. Developing cognitive user models will assist in creating agents that help rather than hinder human performance. Because this will be an evolving system, maintaining an accurate user model that evolves with the system will facilitate system design, interface design, and development of current training materials. Given such a model, it is possible that updating portions of the training materials, such as the steps for completing system tasks, can be mostly automated.

Understanding the commander's interface requirements will be done using a combination of task analysis techniques. Use cases will be developed to capture the overall system goals, who will be using it, and how we expect it to be used. These use cases will change throughout the project, but a common reference point is necessary for the development team. A Job Analysis will be conducted for each of the user groups (e.g., commanders, platoon leaders, controllers, etc.) to understand how the system will fit within the scope of their overall jobs. A GOMS Analysis (Card, Moran, & Newell, 1983; Kieras, 1998) will also be conducted for the user tasks that will be affected by the system, such as communication procedures.

The GOMS family of techniques is among the best developed engineering methods for modeling human performance with computer systems (John & Kieras, 1996). The standard GOMS model is based on a standard information-processing model of human performance and the resulting models can be used to determine information requirements, functional coverage, execution and learning time, and overall interface consistency. In addition, GOMS models can be used to make qualitative predictions about where users will make errors (Wood, 1999). These GOMS analyses will clarify which portions of the user tasks can best be enhanced by the system and to what degree they will be improved. In addition, GOMS models provide an excellent framework with which to build training materials and system documentation (Elkerton & Palmiter, 1991) because they specify the procedural and declarative knowledge necessary to perform the modeled task. This knowledge is organized in discrete procedures and all decisions and any complex cognitive operations are clearly indicated.

Phase I System Environment

The 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.

Integration Environment: Operator Control Unit / OneSAF TestBed

Bialczak, Nida, et al. (Science and Engineering Services, Inc., 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.

Because the OCU was to play an integral role in the exercises at the MMBL, the OneSAF Test Bed Semi-Automated Forces (OTBSAF) simulation tool was chosen to provide a foundation for the OCU over actual existing control units. This provided an interface to the Distributed Interactive Simulation (DIS) network used at the MMBL. Creating different types of unmanned vehicles including scout robots, main battle robots, missile robots, mortar robots, rocket robots, mule robots, resupply robots, towing robots, mine-laying robots, counter-mine robots, and all sizes of Rotary Wing Aircraft (RWA) and Fixed Wing Aircraft (FWA) entities in the OCU was similar to creating other OTBSAF vehicles.

Many unmanned vehicle control interfaces are too complex to be useful on the battlefield. To simplify the OCU, but still maintain flexibility, default parameters can be established at the start of an exercise through a pop up window. To task a robot, the operator selects the appropriate task and specifies its unknown parameters. For example, to give a Micro Air Vehicle (MAV) a hover command, the operator selects *hover* from the list of tasks for that robot, and clicks on the map to designate the MAVs orientation. The altitude of the hover task is set in the default parameters window. If a change is necessary, the operator can manually pop up the window to change the defaults. This simple but flexible tasking mechanism allows the soldier to concentrate on the mission at hand instead of details on how to task the robots.

A key feature of the OCU is switching control of the unmanned platforms from one OCU to another. This is required because an operator may want to relinquish control of some of his unmanned vehicles for various reasons (perhaps a heavy fire-fight) or the OCU vehicle itself could be destroyed. With two mouse clicks, the operator can send a Disconnect message to a robot, then with a single mouse click an operator at another OCU can send a Connect message to take control of that robot. If an operator attempts to control a vehicle that is already under control, a warning message appears on both the OCUs. If an unmanned vehicle has not received any messages from its OCU for a designated time, the robot sends an *Uncontrolled* message to the nearest OCU with communication range. Any OCU within range can take control of the robot.

Another feature of the OCU is an adaptable fire control. Armed robots have active and reactive fire control. The active fire control is dependent on the target's acquisition level. A human's or agent's knowledge about a target has several levels: detected (see something, but do not know what it is), classify (it is tracked, wheeled, etc.), recognize (it is a tank, APC, etc.), and identify (it is a T80, T72, etc.). Active fire control has all these levels plus a Fire on Order level to provide a man-in-the-loop option. With the active fire control set at identify, the unmanned vehicle will not shoot until it knows its target is a T72, for example. On the other hand with active fire control set on *detect*, the robot will fire at any target it sees. Maintaining BLUFOR situational awareness should prevent friendly fire. The reactive fire control is either hold fire or return fire if fired upon. This fire control mechanism together with target images gives optimal control of the unmanned vehicle's armament to the soldier.

The OCU can receive images from smart and not-so-smart unmanned platforms. Some robots have the ability to recognize possible targets. When a target is acquired, an icon appears on the OCU map and an image is sent to the OCU. These images come in three formats, small chip, large chip, and complete scene. The small chip contains the least amount of information. It is cropped around the target, and some of the pixel data has been filtered out to reduce transmission bandwidth requirements. It appears on the OCU with a prompt to request the large chip or not. If the operator can distinguish the target from the small chip, the operator dismisses the prompt– otherwise the robot sends the large chip. This chip contains more information, less cropping is done, and more pixels are kept. Again, a prompt is given for the entire scene. If this prompt is selected, the robot will send the original image to the OCU. At any phase of this process, the operator can dismiss further images and if warranted, take action with armed platforms. Some of the smaller unmanned air vehicles may not have the ability to identify targets. Images from these vehicles are periodically updated on a screen next to the OCU. The views from up to four unmanned air vehicles can be displayed at once. These realistic images of the viewpoints from the robots make the simulation more believable for the soldiers at the OCU.

The OCU can also control *real* unmanned vehicles. Currently, the OCU is capable of controlling a 350-pound electric driven wheeled robot, a 650-pound diesel powered tracked robot, a 50-pound electric driven wheeled robot, and a 5-pound electric powered wheeled robot. Several scenarios have been created with simulated and real robots from the OCU. One consisted of the 50-pound robot being tasked from the OCU to recon a built up area in the simulation. As the real robot moved across a parking lot, its icon on the OCU responded to its movements around the town. A MAV was tasked to track the robot and an IG provided realistic imagery from the unmanned air vehicle.

In summary, 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.

Agent Environment: The Soar Cognitive Architecture

The Soar cognitive architecture is a powerful framework for creating multi-agent systems. Soar has been used successfully to simulate complex human behaviors, the most extensive of which is Tac-Air-Soar (Jones, et al., 1999). The interfaces we have developed for

TacAir-Soar allow one or two operators to control more than 50 autonomous agents during training exercises.

Soar is a common software architecture that has been used to model a wide range of complex psychological behaviors (Rosenbloom, Laird, & Newell, 1993). It was originally used to develop psychological models and intelligent systems that require specific problem-solving methods from many different domains. The Soar architecture has since evolved to include representations and methods for problem solving, planning, learning, and interacting in complex, dynamic environments. Many of the design requirements that contributed to Soar were derived from Newell and Simon's work on modeling human problem solving (Newell & Simon, 1972). Based on initial successes in modeling human behavior, Newell proposed Soar as a candidate "Unified Theory of Cognition" (Newell, 1990).

All Soar models share the same memory structure, task decomposition, task processing, and learning structure. Different systems developed within Soar have successfully modeled a wide variety of human behavior relevant to this research.

A key component of all Soar models is that all activity is cast as a succession of decisions as to what to do next. The decisions are based on an internal representation of the current situation, which is based upon realistic simulated sensors (such as simulated radar, visual, IFF, RWR, FLIR, TVS, etc.).

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). The current project uses Soar to develop Intelligent User Interface agents, leveraging Soar to support a rich and reactive human-like task decomposition, but without being limited by human performance constraints.

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 & Singh, 1997) has defined an agent communication language 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 also 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. This will make it more difficult for enemy forces to compromise robotic force communications.

User Interface Development: Tcl

Tcl is a leading scripting language supporting robust and rapid GUI development that can easily be modified and ported across platforms. Tcl was selected for the Phase I interface prototype due to its high quality and low integration cost. Interface prototypes are necessary for both testing our design assumptions with real users (or other subject matter experts), and for stress-testing the underlying technologies during system development. Soar Technology has successfully created similar interfaces for controlling simulated semi-autonomous agents for flight training within the TacAir-Soar system (Jones, et al., 1999). The resulting interfaces allowed the migration from a system that required one human for every five vehicles to a system where one to two people can control an entire exercise (forty or more vehicles). Similar to the needs of the current project, the TacAir-Soar system required planning and control components for air tasking orders, team assignments, and individual entity assignments.

Phase I Activities and Accomplishments

Under Phase I of this Small Business Innovative Research contract, we researched methods for effectively controlling FCS units containing mixed human and robotic elements. Our objective was to determine whether an agent framework built around three specified agent types (Tasking, Coordinating, and Monitoring) could be constructed to add an intelligent abstraction layer between human commanders and battlefield elements. The focus was to identify human-system interaction issues, design potential solutions, and create software that supports the commander's tasks and mitigates inherent human performance limitations.

During the initial phase of this project we progressed in several areas. Major accomplishments of Phase I included:

- A working scenario was defined on which to determine project feasibility.
- Architectural tradeoffs were discussed culminating in a working system and communication architecture.
- Necessary inter-agent communications protocols to perform the scenario were designed.
- Soar was integrated into the Operator Control Unit (OCU)/OneSAF TestBed (OTB) code-base.
- A Soar-language prototype multi-agent C^3I support system was developed and tested.
- A prototype user interface for the multi-agent system was designed and implemented.

Conceptual FCS Vignette

We chose a simple scenario of a UAV acquiring a target for an unmanned shooter. This scenario was then split into several stages and the responsibilities for individual entities were identified. This scenario is illustrated in Figure 3.

1. Commander wants to attack enemy unit with non-line-of-sight (NLOS) weapon.
2. Commander selects unit and drags a path to the enemy unit.
3. Tasking agent interprets the gesture as an attack, completes 80% of an operation order and queries the commander for the rest of the details.
4. Commander confirms the op order and tasking agent translates the order into a set of smaller movement orders.
5. Tasking agent spawns a Monitoring agent that gets unit status every 5 minutes (or after anomalous event) and updates commander's screen.
6. Monitoring agent informs commander that unit is ready to fire.
7. Commander approves, unit fires, sensor unit informs coordinating agent that enemy is destroyed.

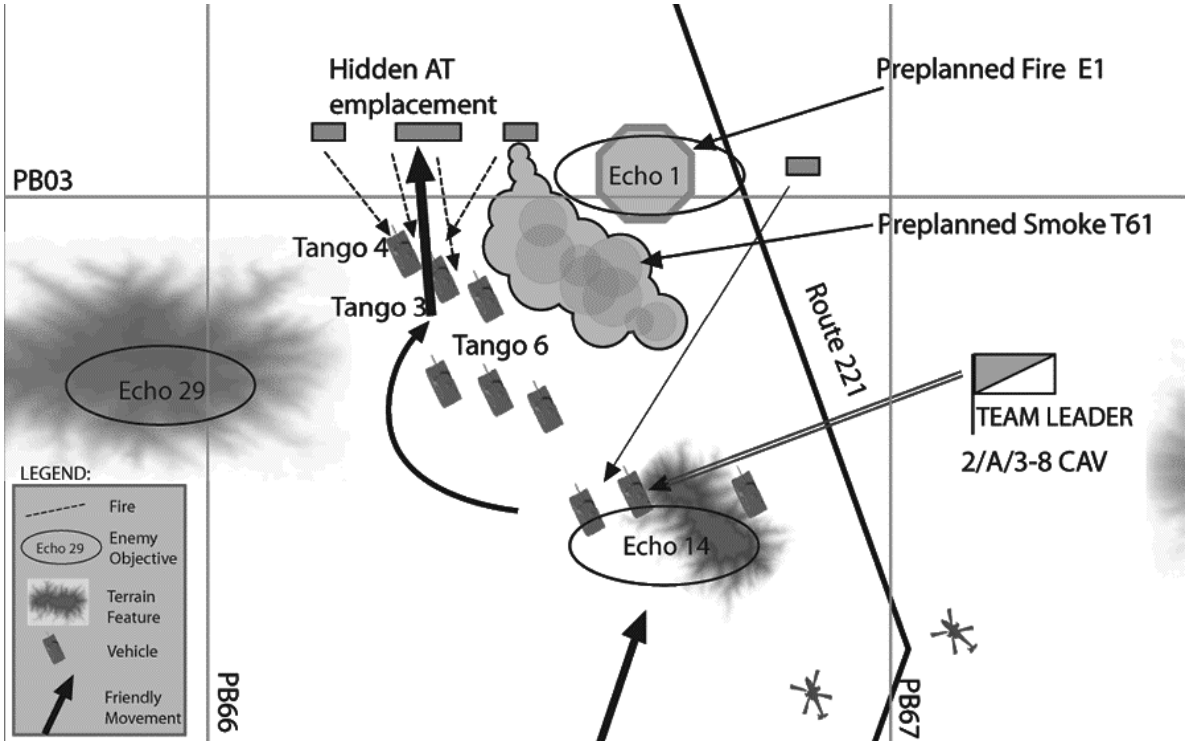


Figure 3. Phase I Scenario Narrative Map.

Phase I Scenario

A scenario was created from the FCS vignette that:

- Would exercise each of the candidate agent types,
- Was simple enough to implement within the scope of the project, and
- Was sufficiently complex so as to adequately explore the feasibility of the approach.

Briefly, the scenario involves an FCS commander targeting an enemy unit with a non-line-of-sight (NLOS) unmanned ground vehicle (UGV). The tasking agent finds an available unmanned aerial vehicle (UAV) capable of providing targeting information to the UGV at the designated attack time. The UGV moves into position and notifies the UAV that it is ready to fire. At this time the UAV paints (lases) the target, allowing the UGV to fire. Table 1 shows the discrete phases in this scenario and describes the responsibilities of each of the involved entities.

Table 1
Scenario phases and entity responsibilities

Event/Agent	Task	Monitor	Coord	Entity/Robot
FIPA Init	Register DF/AMS	Register DF/AMS	Register DF/AMS	Register DF/AMS
FIPA Connect	Search DF for "local" Mon/Coord	Search DF for "local" Task/Coord	Search DF for "local" Task/Mon	
Commander Needs Attack (1)				
Commander Designates Attack (2)				
Attack Order Interpretation (3)	Interpret Order: auto-fills slots, queries for Cmdr for details, queries Coord for immediately accessible details	Fills in whatever details it knows about at this time from COP		
Determine Resources (3)	Find required entity resources from Coord agent		Seeks entity resources: Search DF for agents e.g. w/ required sensors that can be in required area at required time with necessary munitions, etc. and reply to Task agent	
Confirm Order (4, 6)	Translate order into primitive behaviors			
	Tasks UAV			Committed
	Tasks NLOS			Committed
	Tells Monitor to watch for task-specific conditions	Sends out commands to sensor entities and to Coord agents to send it info and under what conditions	Receive monitoring tasks and searches and dispatches requests to appropriate other commander/TCMs (with replies directly back to Monitor)	Receives monitoring tasks

Table Continues

Monitor Execution (5)	Receives periodic status, passes on to Task, and informs and presents current operational picture to Cmdr		Sends periodic status
	Receives events, passes on to Task, and informs and presents current operational picture to Cmdr		Reports events
Prepared To Fire (7)	Receives ready to fire		NLOS: Reports ready to fire
	Receives marking target		UAV: Reports marking target
FIRE! (8)		Tells entity to pull trigger	NLOS FIRES
Target Destroyed (8)	Receives kill report		UAV reports kill
	Informs Cmdr and Task agent of success		

List of Scenario Phases

(1) Commander wants to attack enemy unit with non-line-of-sight (NLOS) weapon.
(2) Commander selects unit and drags a path to the enemy unit.
(3) Tasking agent interprets the gesture as an attack, completes 80% of an operation order and queries the commander for the rest of the details.
(4) Commander confirms the op order and tasking agent translates the order into a set of smaller movement orders.
(5) Monitoring agent gets unit status every 5 minutes (or on anomalous event) and updates commander's screen.
(6) Tasking agent knows that unit will require targeting info when it gets into range, so it contacts the Coordinating agent which requests services of a sensor unit at time T. (OBE)
(7) Monitoring agent informs commander that unit is ready to fire.
(8) Commander approves, unit fires, sensor unit informs coordinating agent that enemy is destroyed.

System and Communication Architecture

Several system design architectures were examined to determine tradeoffs. It was decided that the most flexible and extensible architecture would be beyond the scope of this phase of the project. This architecture would entail discrete system components and a dedicated communications infrastructure. It was determined that this portion of the design was merely a matter of engineering and did not affect the feasibility questions being addressed. Instead, a simpler architecture was proposed similar to Soar Technology's prior work with JSAF, whereby the Soar kernel is compiled with the OCU/OTB code-base. This simplifies the inter-agent communications infrastructure and allows more time to be spent on agent design issues. The following sections discuss these tradeoffs in more detail. Four options were considered and Option III was selected for Phase I implementation. This architecture will be reevaluated for Phase II. All four options are presented below.

- Features
  - Soar agent types are: Tasking, Coordinating, Monitoring, Entity
- Pros
  - Don't need to build an additional FIPA-OCU interface component
- Cons
  - Medium number of Soar agent types required
  - Reduces composability with entity agent types (taskframes, etc)

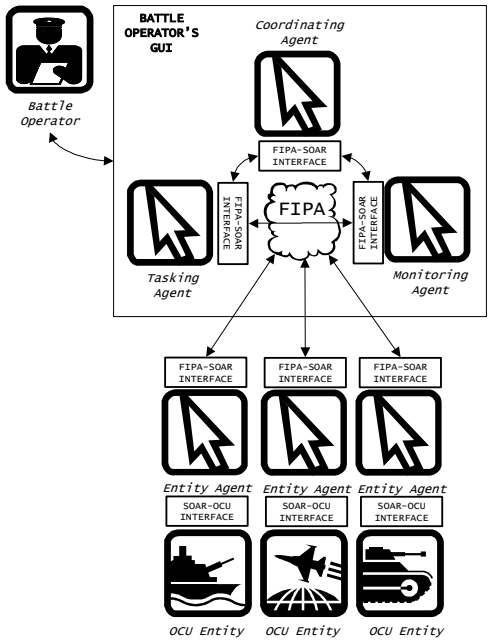


Figure 4. Phase I Design Option I.

Option 1 (Figure 4) would have had the advantage of being able to apply the type of complex pro- and re-active cognitive modeling of Soar architecture agents to every entity in the OCU. The disadvantage would be that much of the ability for the Tasking, Coordinating, and Monitoring agents to engage in "plug and play" composability with low-fidelity agents would be removed by virtue of having to implement a Soar agent wrapper around every new low-fidelity entity.

- Features
  - Soar agent types are: Tasking, Coordinating, Monitoring, Team, Entity
- Pros
  - Don't need to build an additional FIPA-OCU interface component
  - Incorporates NCO cognate as Team Agent in parallel to C^3 cognates of Tasking, Coordinating, Monitoring
- Cons
  - Largest number of Soar agent types required
  - Seriously reduces composability with entity agent types (taskframes, etc)

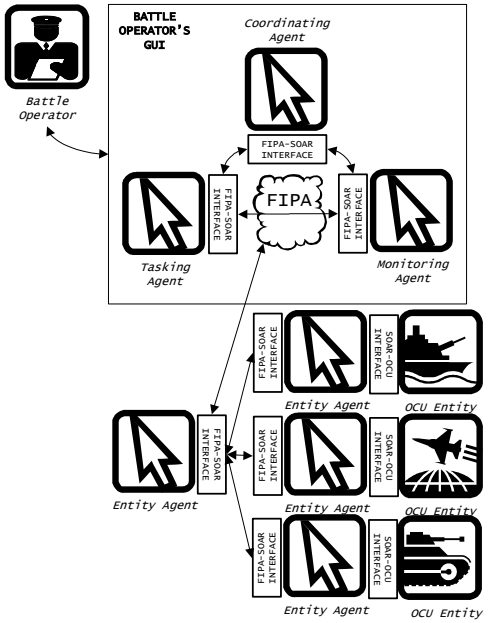


Figure 5. Phase I Design Option II .

Option II (Figure 5) was essentially a modification of Option I which was discussed that would add another layer of coordination between the C3 agents and collections of entities grouped as "Teams." Each team would have a Team Agent essentially acting as the NCO and mediating between the entities and the C3 agents (Tasking, Coordinating, Monitoring). Although this option suffered from the same disadvantages as the first, the Team Agent was an interesting addition.

- Features
  - Soar agent types are: Tasking, Coordinating, Monitoring
  - Selected as Phase I design paradigm
- Pros
  - Smallest number of Soar agent types required
  - Greatest degree of composability with entity agents (taskframes)
- Cons
  - Need to build an additional FIPA-OCU interface component

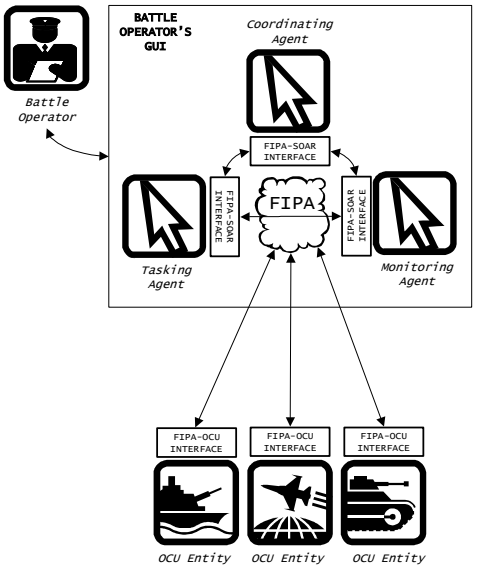


Figure 6. Phase I Design Option III.

Option III (Figure 6) required the development of an interface between OCU entities and the FIPA messaging layer in addition to the required interface between Soar agents and the FIPA messaging layer, the preservation of relatively easy composability with arbitrary entity types was deemed more than sufficient to outweigh that additional development cost. For that reason, this option was deemed the best choice for the Phase I implementation.

- Features
  - Soar agent types are: Tasking, Coordinating, Monitoring, Team
- Pros
  - Don't need to build an additional FIPA-OCU interface component
  - Incorporates NCO cognate as Team Agent in parallel to C^3 cognates of Tasking, Coordinating, Monitoring
  - Team agent is sole point of update for additional entity types
- Cons
  - Team agent needs to be updated for any additional entity types

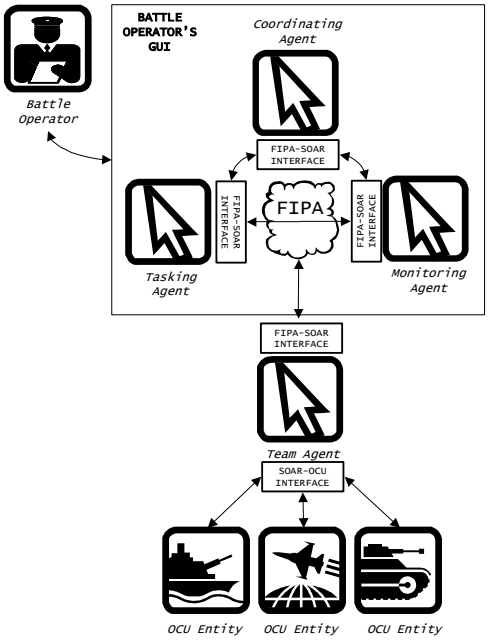


Figure 7. Phase I Design Option IV.

Option IV (Figure 7), much like Option II, was essentially a modification of Option III that would add another layer of coordination between the C3 agents and collections of entities grouped as "Teams." Each "Team" would have a Team Agent essentially acting as the NCO and mediating between the entities and the C3 agents (Tasking, Coordinating, Monitoring). Although this option offered potential merit above and beyond the design of Option III, the relative advantages did not seem as though they could be cost-effectively presented in the Phase I implementation.

Inter-Agent Communications for the Scenario

The inter-agent communications necessary to perform the scenario were diagramed fully to more clearly delineate the roles and responsibilities of all the scenario entities. Figure 8 provides a sample portion of the communications diagram developed to illustrate the basic flow of communications among scenario entities. This figure's intent is to provide the reader a pictorial representation of the output format from the analysis. Output content for this sample can be found in Appendix B.

The inter-agent communications necessary to perform the scenario were diagramed fully to more clearly delineate the roles and responsibilities of all of the scenario entities. This was translated into the UML (Unified Modeling Language) sequence diagram shown in Figure 8 and presented in full in Appedix B. This form of diagram formalizes the timing and sequence of events necessary for the agents to cooperate for the completion of the scenario task.

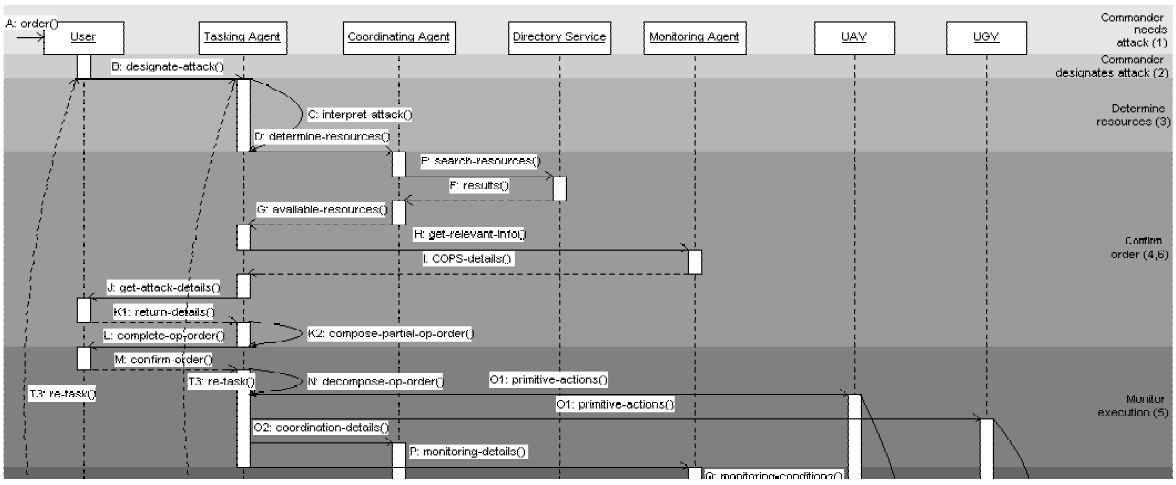


Figure 8. Inter-agent communication partial diagram (See Appendix B for complete diagram)

Soar-OCU/OTB Integration

The Soar cognitive architecture has been successfully integrated with the OCU/OTB code-base and Soar agents are currently able to see the status of OCU entities and send commands to those entities. While there were minor technical difficulties during the integration process, the system is currently quite stable and should provide a good platform for future work. Considering that the OCU/OTB codebase is essentially itself a prototype, the integration went quite well. With this portion of the project complete the stage was set to allow the interface agents to send commands to and receive messages from the OCU entities.

Soar Prototype Agent

A Soar prototype agent was created to test the feasibility of the Soar agent to OCU entity communication design and the overall system design. The complete functional path of the system; task specification by the user through the user interface, task interpretation and elaboration by the agent, and task performance by the OCU entities was tested successfully using the scenario described earlier.

User Interface Prototype

A prototype user interface for the multi-agent system was designed and is shown in Figures 9 and 10. The interface consists of a multi-tabbed pane that contains controls for initiating, monitoring and reviewing the progress of the scenario task described earlier. This interface is intended for initial testing purposes only and is not meant to represent a final user interface design. Its main purpose is to allow rapid prototyping and testing of user interface concepts associated with initiating, monitoring and reviewing the progress of the task. This interface is used in conjunction with the main OTB/OCU interface, which already provides the ability to override low-level commands, and provides a good picture of the current situational awareness of the OCU entities.

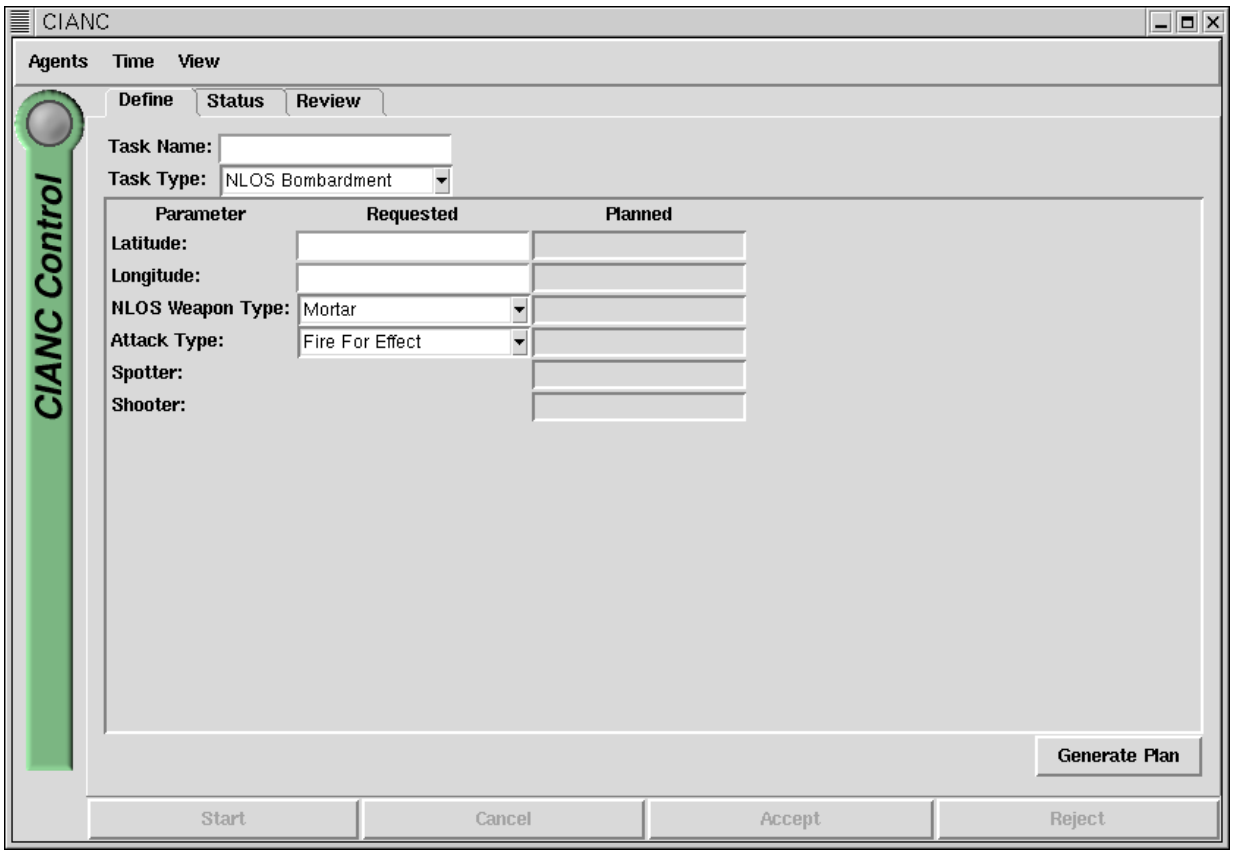


Figure 9. The Task Definition Pane of the user interface prototype.

Figure 9 displays the Task Definition pane of the prototype user interface. The top of this pane contains fields for naming the task and selecting the task type. Currently the straw man task is the only valid task type. The task naming fields will be used for task management purposes in future tests involving multiple concurrent tasks. The central portion of this pane contains the plan specification fields that must be filled for the mission along with available optional parameters that may be specified by the user if desired. The "Generate Plan" buttons allows the user to submit the plan request to the tasking agent. The plan created by the tasking agent is then displayed along side the plan specification fields to allow the user to easily compare the generated plan parameters with the requested plan parameters. Future versions of this interface will also provide the user with explanations for any differences between the generated and initially requested plans. These tasking plans extend the current capacity of the integrated CIANC^3-OCU system by providing the warfighter a limited degree of mission planning. Future versions will extend the scope of mission planning available

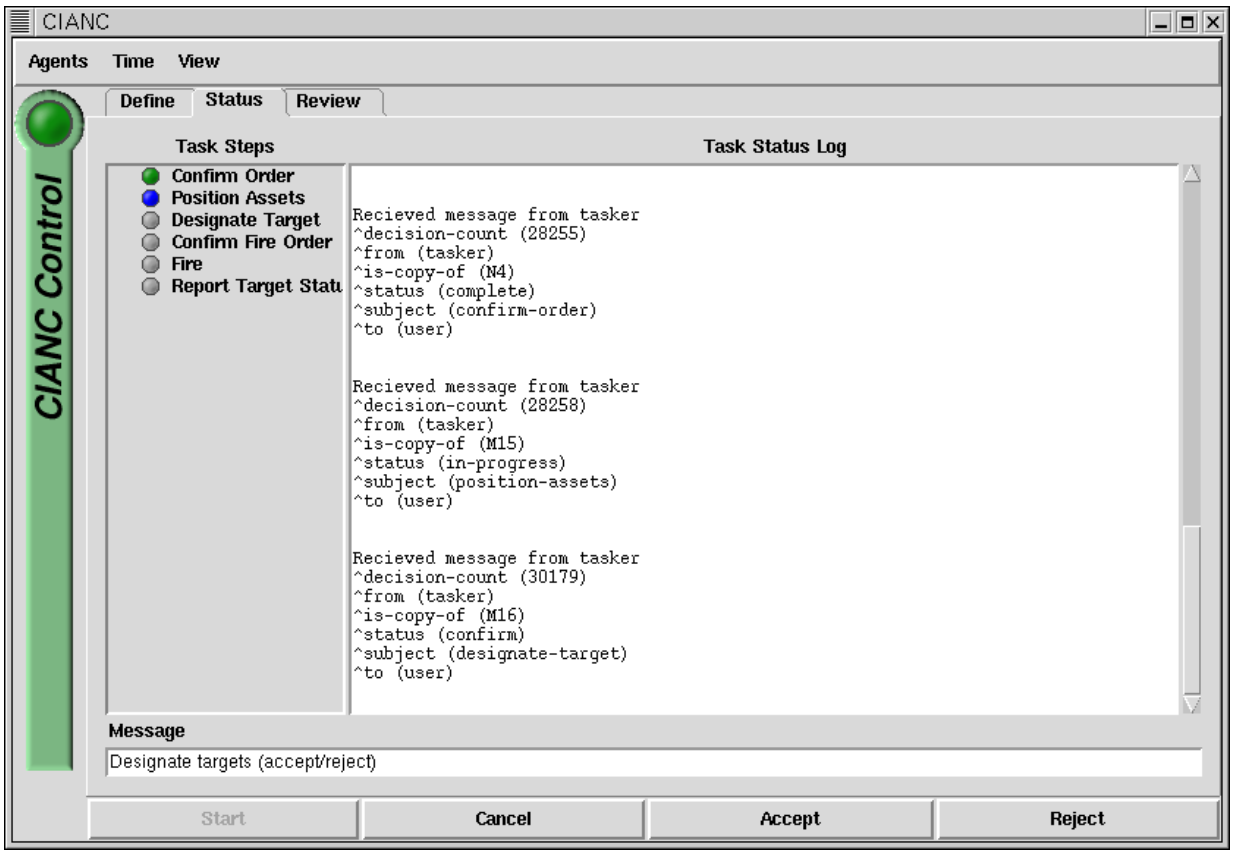


Figure 10. The Task Monitoring Pane of the user interface prototype.

The Task Monitoring Pane is shown in Figure 10. This pane is intended to provide the user with a simple and easy to understand interface for determining the current status of the task. It also provides facilities for the user to respond to queries from the agents (permission to fire, etc.). The overall task status is displayed with an indicator that is visible no matter what pane is currently being displayed in the user interface. This status indicator is located in the upper left hand corner of the interface and uses colors and flashing to provide the user with status information.

CIANC Agent Behavior in the NLOS Bombardment Mission

In Phase I we built a demonstration prototype of the CIANC system to perform the scenario described earlier. This section details the behaviors and communications within the system as it is performing the scenario.

When an agent is started, it is initialized with an agent type (Tasking, Monitoring, Coordinating) that determines how it will apply its knowledge to the environment. A Tasking agent will immediately request direction from the operator and wait for a response, while a Monitoring agent will examine the environment to determine and report what the Common Relevant Operational Picture (CROP) is.

When the Tasking Agent has been communicated a mission definition by the operator, it will request CROP details from the monitoring agent, including a report of available FCS vehicle resources. From that list of available resources, the Tasking Agent will attempt to compose a mission operational plan based on the requirements of the mission and the preferences of the operator. Once the Tasking Agent has determined whether or not it is possible to compose a functional mission plan, it will report to the operator. If sufficient information and resources were available for the Tasking Agent to compose an operational plan, it will present the plan for acceptance to the operator, otherwise, the Tasking Agent will indicate the deficient information and/or resources to the operator.

If the operator communicates to the Tasking Agent that it can start the plan, the Tasking Agent will direct its selected Spotter entity to begin proceeding to the defined contact area. The Monitoring Agent will continue monitoring the status of selected entities and the environment, reporting relevant changes to the Tasking and Coordinating Agents. The Coordinating Agent will report to the Tasking Agent when mission-specific milestone synchronization events have occurred, allowing the Tasking agent to further direct selected entities and report to or request feedback from the operator.

When the Monitoring Agent reports to the Coordinating Agent that the selected Spotter entity has detected targets according to the defined Rules of Engagement (ROE), the Tasking Agent will direct the Spotter entity to halt and continue observation of the targets. The Coordinating Agent, based on CROP reports from the Monitoring Agent, will then inform the Tasking Agent that a mission milestone, the Spotter entity in position observing the targets, has been achieved. If any CROP reports from the Monitoring Agent indicate that the pre-requisites for that mission milestone are no longer met, such as the Monitoring Agent reporting that the Spotter entity has lost sight of the targets, the Coordinating Agent will inform the Tasking Agent of the change in mission milestone status. In the example of the Spotter entity losing sight of the targets, the Tasking Agent will then direct the Spotter entity to re-acquire visual contact with the targets.

As long as the Tasking Agent believes that the Spotter entity is in position and is observing the targets, if the Monitoring Agent indicates that the Shooter entity is beyond indirect fire weapon range of the targets, the Tasking Agent will command the Shooter entity to advance towards the contact area until within indirect fire range of the targets.

Once the Monitoring Agent reports that the Shooter entity is within indirect fire range of the targets and the Coordination Agent has not informed the Tasking Agent of any change in milestone status, the Tasking Agent will direct the Shooter entity to halt in preparation for fire. The Coordinating Agent, based on CROP reports from the Monitoring Agent, will then inform the Tasking Agent that another mission milestone, a combination of the Shooter entity in firing position and the continuing Spotter entity observation milestone, has been achieved.

The Tasking Agent will recognize these as the pre-requisites of the target designation portion of the NLOS Bombardment mission and will request authorization from the operator to begin actively designating the targets. The Tasking agent will wait until the operator replies with an acceptance message or the situation otherwise changes. When the Tasking Agent receives

operator authorization to designate the targets, it will direct the Spotter entity to begin target designation. The Coordinating Agent, based on CROP reports from the Monitoring Agent, will then inform the Tasking Agent that another mission milestone, both selected entities in position and the Spotter entity designating the targets, has been achieved.

With the Coordinating Agent reporting the existence of the pre-requisites for the indirect fire portion of the NLOS Bombardment mission, the Tasking Agent will request authorization from the operator to begin firing upon the target position. The Tasking agent will wait until the operator replies with an acceptance message or the situation otherwise changes. When the Tasking Agent receives operator authorization to fire upon the targets, it will direct the Shooter entity to begin bombardment of the target location as reported to the Monitoring Agent by the Spotter entity.

The Monitoring Agent will continue to report the status of the targets as observed by the Spotter entity. As long as the Tasking Agent believes that it has fire authorization from the operator, that the Spotter entity is observing the targets, that the Shooter entity has remaining ammunition, and the targets are not destroyed, it will direct the Shooter entity to fire short barrages.

Once the Monitoring Agent reports to the Coordinating Agent that the Spotter entity has observed the destruction of the targets, the Coordinating Agent will indicate to the Tasking Agent that the final mission milestone has been achieved. When the final mission milestone is achieved, the Tasking Agent will report this to the operator, who is then free to re-task the agents and/or the individual entities.

Summary of Phase I Results

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 the Coordinating and Monitoring agents, driving their further elaboration.

The UML sequence diagram 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.

Phase II

Phase II System Design

The purpose of this section is to provide a high level overview of the design of the software infrastructure that will be built for Phase II of the CIANC^3 project.

Infrastructure Design

The Phase I CIANC^3 infrastructure is built upon the Multi-functional Operator Control Unit (OCU) software provided by Science and Engineering Services, Inc (which is in turn based on OneSAF Testbed Baseline Version 1.0 (OTB). The OCU software was chosen as a basis for this project (over OTB) mainly because it already provides the necessary simulated robotic entities for the Soar meta-agents to manipulate. Use of the OCU software will continue in Phase II.

Design Concept

The software infrastructure for the CIANC^3 project can be broken down into three main components: the Soar kernel, responsible for creating and managing the Soar meta-agents; the OCU, responsible for creating and managing the OCU entities; and the Message Manager, which is responsible for the messaging infrastructure between the various agents in the system. The interactions between the Soar kernel, the OCU/OTB and the Message Manager will essentially be limited to FIPA-ACL type messages except for a few interactions during process initialization and agent directory querying.

The Message Manager will act mainly as a message queue and a directory service for facilitating communications between the various agents. The specifics of the message content are described in another document. Here it is enough to say that these messages are the only method by which information is passed between the components of the system. This approach should help reduce unnecessary coupling between the various components while still providing rich interactions between the various agents.

The Messaging Infrastructure

The Message Manager as implemented during the Phase I of the CIANC project was a prototype closely tied to the implementation domain of the Phase I vignette. During Phase II, it is our intention to enhance the Message Manager component of the software infrastructure to be both general purpose and extensible. By enhancing the messaging capability to be both general purpose and extensible, the process of adding new messages in the future and tying them to existing agent behaviors could be done without requiring a behavior developer. This parameterized approach significantly extends the composability of any behaviors created for the tasking, coordinating and monitoring agents. Enhancement of the Message Manager component of the software infrastructure will be obtained by standardizing the messaging software interface between all agents and entities within the OCU and using an Agent Communication Language (ACL) like the COABS Grid or FIPA.

Adoption of COABS formalisms for the composition of messages will allow us to encode general message handling and message composition knowledge within the agents. Specific message instances and their contents can then be parametrically associated with behaviors or sets of behaviors, enabling the possibility of behavior composition by non-behavior developers.

Figure 11 shows in detail the execution sequence for a single messaging cycle. In a single messaging cycle, the agent simply retrieves messages from his message queue using the Retrieve Messages call and sends them using the Send Message call. The OTB scheduler calls both the agents' and OCU entities' messaging loops indirectly through their main loop cycles. The Soar agent's main loop in turn calls the input and output phase callbacks, which are directly responsible for handling the message sending and retrieving.

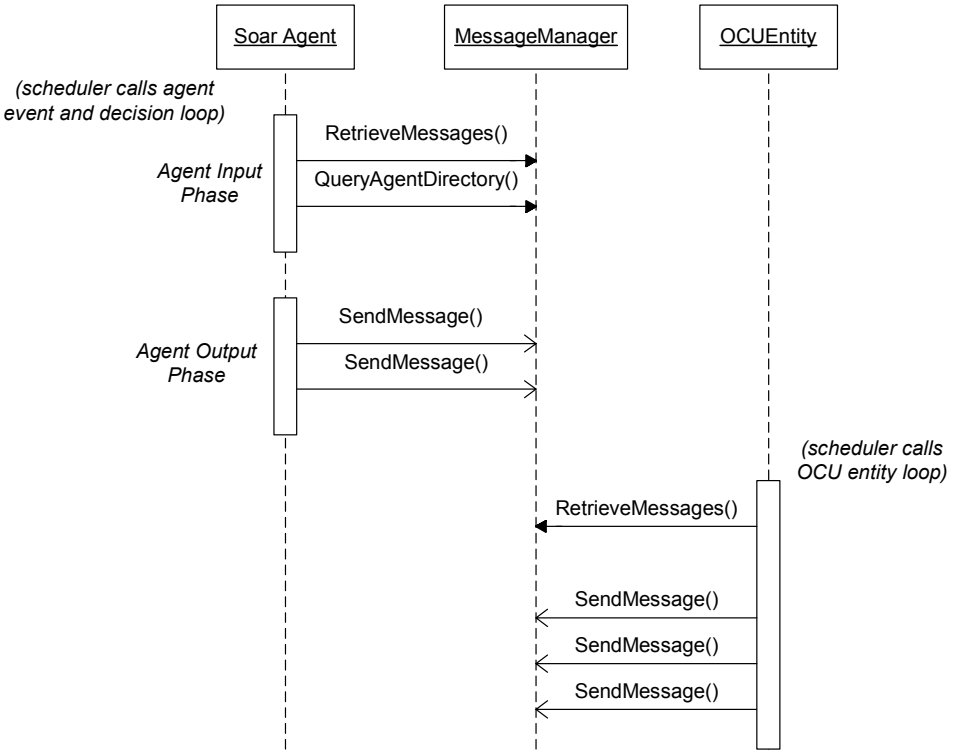


Figure 11. Messaging sequence diagram.

A higher-level view of the messaging infrastructure, which concentrates on the messaging necessary to maintain the agent database, is shown in Figure 12. Here the agents register with the Message Manager/Directory database on creation and then provided periodic status and capability updates to maintain the database information. A time stamp detailing the time of the last update will be provided as part of any database query result. This should allow the agents to gracefully handle situations where other agents or entities have been destroyed or disconnected.

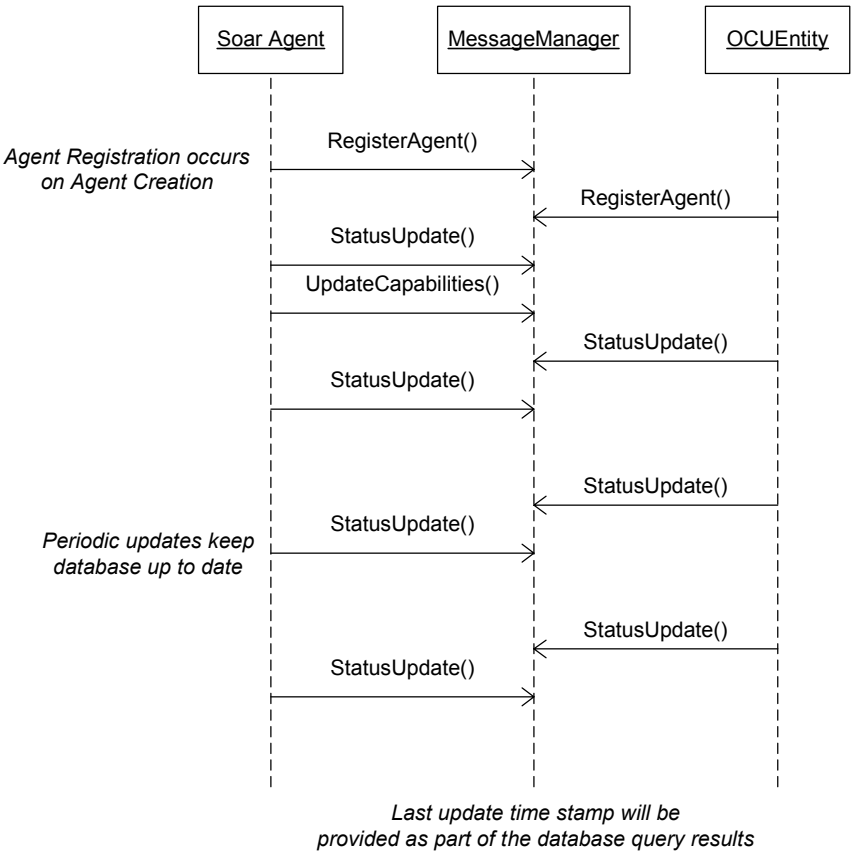


Figure 12. Agent registration and status messaging.

Multi-Agent Design

Following are design requirements for each of the agent types, where each listed item describes a single feature/requirement. Some listed features will require interaction with other agents; these are shown in brackets.

Tasking Agent (TA). Task agents assist commanders and controllers to rapidly issue battlefield commands.

- Present commander with reasonable operation plan.
- Engage commander in clarification dialog.
- Present options to commander.
- Translate order into proper command sequences for next command layer.
- Autonomously gather information to support Op Order completion [Coordination, Monitoring].
- Dispatch and execute Op Order [Coordination, Monitoring].
- Task UAVs [Coordination, Monitoring].

- Autonomously gather information to maintain situational awareness [Coordination, Monitoring].
- Autonomously modify tasking [Coordination, Monitoring].

Coordinating Agent (CA). Coordinating agents are responsible for facilitating communication and coordination across and within echelons within the command hierarchy.

- Develop and maintain representation of available and currently interacting command elements [Monitoring].
- Determine low-level communication attributes such as radio frequency, call signs, unit designations, chain-of-command, IFF and communications security [Monitoring].
- Develop and maintain representation of common operational picture with regard to current situation, plans, enemy intentions, and BDA. [Monitoring].
- Coordinate with higher, lower, and nearby peer command elements to react quickly to changing situations, reduce duplication of effort, maximize target coverage, synchronize attacks, and massing fire [Monitoring].
- Present current operational picture to commander (OBE).
- Set up direct sensor to shoot communications across commands.
- Set up indirect fire cross-command communications.
- Facilitate teleconferencing.
- Reestablish broken communications [Tasking, Monitoring].
- Integrate orphaned units [Monitoring].
- Share incomplete sensor information to upper echelons.
- Decompose and disseminate Op Orders to lower echelons.

Monitoring Agent (MA). Monitoring agents are responsible for assisting the commander in maintaining an accurate awareness of the current situation at all times.

- Assist commander in maintaining accurate awareness of the current situation [Coordinating].
- Fuse, filter, and prioritize raw data.
- Transform data into contextually relevant information [Tasking].
- Autonomously gather information to maintain common operating picture [Tasking, Coordinating].
- Present current/common operational picture to commander.
- Collect and present all messages related to a particular unit to help build a broader picture from single events [Coordinating].
- Represent visually direct communication lines between shooters and sensors [Tasking].
- Monitoring health and stress levels of human subordinates.

Phase II Work Plan

This project will address the robotic command-and-control needs by developing a framework and toolkit for constructing intelligent user interface elements to enhance commander-team control and interaction. These user interface elements will allow battlefield

commanders to efficiently control autonomous teams of robotic or human elements by anticipating commander intent and reducing substantially the amount of information the commander must specify to communicate that intent. It will also support the integration of other intelligent and autonomous decision-support tools and information systems without increasing the commander's overall mental workload.

There are three main research & development areas in the Phase II workplan; human-system interaction, agent & architecture research, and system development. The toolkit will consist of extensible software agents for tasking, monitoring, and coordination, driven by a direct-manipulation user-interface. An initial task analysis of the command and control problem will be used to identify the critical cognitive workload factors. This will motivate the assignment of appropriate human control functions to software agents and will help determine the requirements for the human interface. A suite of software agents will be developed using the Soar cognitive architecture. These agents will communicate using the Foundation for Intelligent Physical Agents (FIPA) communication protocol and will be controlled using a JAVA -based, direct-manipulation interface. The initial system will be integrated into the OCU-enhanced OTB battlefield simulation system created by TSA and tested with a representative FCS-company scale ground-force task. The following sections describe the tasks that will be accomplished in Phase II.

Research HSI, Scenarios, Behaviors

A variety of methods will be used for assessing the knowledge, skills, and abilities required by a commander to use the system successfully, including observation of commander performance during an FCS C2 exercise and hierarchical task modeling of CIANC^3 and OCU usage. In addition, we will work with subject matter experts to help develop realistic scenarios and to better understand the challenges that will be faced by future FCS commanders.

Develop GUI

Based on the results gathered from the HSI research, an easy-to-use GUI will be designed and built to facilitate communication and interaction between human and software agents in the system. The Phase II GUI will be based on the GUI prototype developed for Phase I but will be more tightly integrated with the OCU, including using highlighting active entity locations, and using the OCU map as a sketchpad to sketch rough plans. In addition, the Phase I prototype focused on single missions. The Phase II GUI will be extended to support multiple concurrent missions. While the Phase I GUI prototype was implemented in TCL, the Phase II interface will be developed in JAVA.

JAVA supports robust, object oriented GUI development and is the native language of the CoABS Grid. User interface elements were created with JAVA using tools and techniques previously developed at Soar Technology. For example, the Communications Panel (Jones, 1998) is a human interface element for communicating with Soar agents operating in the JSAF (Joint Semi-Automated Forces) environment. Similarly, the Situation Awareness Panel (SAP) (Jones, 1999) also communicates with JSAF, allowing users to inspect the reasoning process of

autonomous pilot agents. The information from the SAP is used to aid observation, use and development of TacAir-Soar intelligent agents.

Design and Conduct System and Human Tests

To determine the efficacy of the approach, it is imperative that the system be tested. These tests could 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 and performance testing of the overall system.

Develop Testing Environment

We will develop techniques for instrumenting the OCU environment to capture accurate millisecond-level human-subject data. This will be used to help evaluate overall system usability and the CIANC^3 system's effect on operator performance, cognitive workload, propensity for error, and affect.



Figure 13. Experimental Environment.

Research Agent Architecture and Communications

We will be developing flexible, declarative analytic interaction models supporting semantically rich analysis processes and human interactions. Interactions will be analyzed for commander's intent and a semantically rich interaction protocol representation optimized to the operational processes will be designed. The result will be a novel approach that tightly integrates message semantics to agent communicative acts and conversation protocols.

Develop Agent Architecture and Communications

Multi-agent organization and deontics research – Analyze organizational features of the system with respect to traditional military roles and more flexible, dynamic, and transient agent teams and adopt/develop appropriate representation and reasoning paradigm. Deontic aspects of obligations, authority, permissions, restrictions, etc. will need to be considered when performing tasks in view of the current and expected organizational structure as well as when planning outgoing communications and responding to incoming messages.

System Development and Integration

Although a minimal software interface was developed for Phase I to integrate the multi-agent system into OCU/OTB, further development will more seamlessly integrate the functionality of the OCU and the CIANC^3 prototype allowing for higher fidelity and more realistic experimentation.

Develop Agent Behaviors

Develop representations and an introspection scheme by which agents can determine an autonomy level appropriate to the current tasks and situation. Agents would adjust how proactive or passive they are with respect to goals from human operators and other agents and how independent they are with respect to other modalities such as beliefs and capabilities. Deontic, organizational, and communication aspects all will potentially impact upon an agent's autonomy level.

Phase III Transitional Plan

This work will provide a technical and theoretical framework with which to further explore interface agents for a variety of command and control tasks. It will demonstrate the power of reasoning systems for user interface development and it will represent the state of the art in interface agent technology.

Phase III Commercialization Strategy

Soar Technology, Inc.'s mission is to create and apply cognitive, knowledge-rich, dynamic systems to fundamentally improve how people work and live. Using multiple artificial intelligence (AI) disciplines, our expertise is in R&D for intelligent autonomous agents, control interfaces, information visualization, advanced human behavior and error-modeling technology.

Our core team developed TacAirSoar, the largest and highest fidelity expert system deployed in military simulation, with a suite of agent and exercise management tools. With one of the strongest concentrations of expertise and experience in the field of military AI, we are at the high end of human behavior representation for military simulation. We are now using the expertise and insights gained in simulation and multi-agent interaction to move into new domains of knowledge management, decision-making and control. The work being undertaken under Phase I, and proposed for Phase II, leverages past, current and prospective work, as our technology evolves.

Though less than five years old, Soar Technology has successfully grown its R&D engineering and consulting revenue at an average rate of 100% per year, commercializing its artificial intelligence and human-computer interaction expertise through contracts for DARPA, Army, Navy, JFCOM, ONR, AFRL, ARI and the intelligence community, and with Lockheed Martin, Raytheon, L-3 Comm, Veridian, IITRI, IDA, MSIAC, and others.

If successful, the technical and theoretical accomplishments produced by this CIANC^3 project could be essential for successfully implementing the Army's FCS force transformation vision because they would enable the multiplier effects necessary for optimal control of robotic teams. In terms of immediate markets, FCS command and control programs will thus be the primary commercial opportunities. During Phase II, discussions will be initiated with the objective of integrating this technology into FCS candidate systems under licensing agreements. In addition, through other potential work with DARPA (for example, the Unmanned Combat Armed Rotorcraft (UCAR) program, or the Mixed Initiative Control of Automa-Teams (MICA) research program), the Army and STRICOM, we can leverage and integrate the technology developed in Phase II for the C2 requirements of FCS. In addition to the FCS, many other future unmanned systems will require similar command and control capability. With Congress' goal that within 10 years, one-third of DOD's deep-strike aircraft will be UAV systems, and within 15 years, one-third of all ground combat vehicles will be unmanned, significant acquisition programs will be underway. Soar Technology is monitoring the participants in this market sector, attending conferences, making presentations to government customers, and fielding inquiries from other technology companies who have inquired about Soar Technology's expert system and intelligent agents and interface technology.

In addition, the interface agent system developed for this project can have applications in other domains with similar command and control requirements. For situations such as natural disasters, riot control, event or personal security and terrorist attacks where tight coordination of multiple cooperating teams is crucial, crisis management is a key domain in which CIANC^3 technology can provide enhanced performance. Other domains include factory control and automation, mass transit management and emergency room management.

To successfully commercialize the results of this proposed research, during Phase II Soar Technology will invest business development effort in three parallel paths: (1) integration into the FCS, (2) other military systems requiring C2 for multiple arrays of robotic entities, and (3) commercial markets. The timeline for these activities is projected below.

For ultimate system integration and fielded applications, both military and commercial, our general strategy is to team, via licensing or joint venture partnerships, with larger companies that have deep domain knowledge, marketing, manufacturing and support capability. While Soar Technology's management has considerable experience in generating business, marketing and finance plans, it is our strategy to offer a unique, key part of the technology solution, not a full vertical market solution. In terms of revenue from commercialization, we anticipate a combination of development funding during system development, royalties from procurement, and software and behavior maintenance/upgrade/support contracts after systems are fielded.

References

Birmingham, W. P., D'Ambrosio, J. G., Darr, T., & Durfee, E. H. (1994). Coordinating decision making in large organizations. *Working Notes of the AAAI Spring Symposium on Computational Organization Design*.

Bradshaw, J. M., Dutfield, S., Benoit, P., & Wooley, J. D. (1997). KAoS: Toward an industrial-strength open distributed agent architecture. In J. M. Bradshaw (Ed.), *Software Agents*: AAAI/MIT.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum.

Cohen, P. R., & Levesque, H. J. (1990). *Performatives in a rationally based Speech Act theory.* Paper presented at the 28th Annual Meeting of the Association for Computational Linguistics.

Corkill, D. D. (1982). *A framework for organizational self-design in distributed problem solving networks.* Unpublished doctoral dissertation, University of Massachusetts, Amherst.

Defense Advanced Research Projects Agency. (2000). *DARPATech 2000 Symposium*, Dallas, TX.

Defense Advanced Research Projects Agency. (2001). *Future Combat Systems Solicitation*. DARPA. Available: http://www.darpa.mil/fcs/Solicit.html [July, 2001].

Defense Advanced Research Projects Agency BAA#01-029. (2001). *Mixed Initiative Control of Automa-teams (MICA) solicitation*. Available: http://dtsn.darpa.mil/ixo/mica%2Easp

Decker, K., & Lesser, V. (June, 1995). *Designing a family of coordination mechanisms.* Paper presented at the First International Conference on Multi-Agent Systems (ICMAS-95).

Durfee, E. H., Kenny, P. G., & Kluge, K. C. (1998). Integrated permission planning and execution for unmanned ground vehicles. *Autonomous Robots, 5*, 97-110.

Elkerton, J, & Palmiter, S. (1991). Designing help using a GOMS model: An Information Retrieval evaluation. Human Factors 33(2), 185-204.

Ferguson, I. A. (1992). *Touring Machines: An architecture for dynamic, rational mobile agents.* Unpublished doctoral dissertation, University of Cambridge, Cambridge, UK.

Foundation for Intelligent Physical Agents. (2000). *FIPA ACL Specification*, [Web Document]. FIPA. Available: http://www.fipa.org [2000].

Fox, M. (1979). Organization structuring: Designing large complex software, (Tech. Rep. No. 9-155). Pittsburgh, PA: Carnegie-Mellon University, School of Computer Science.

Fox, M. S. (1988). An organizational view of distributed systems. IEEE Transactions on Systems, Man and Cybernetics, 11(1), 70-80.

Gasser, L., & Hill, R. W. (1990). *Engineering Coordinated Problem Solvers* (Vol. 4). Palo Alto, CA: Annual Reviews, Inc.

Gat, E. (1992). *Integrating planning and acting in a heterogeneous asynchronous architecture for controlling real-world mobile robots.* Paper presented at the Tenth national Conference on Artificial Intelligence, San Jose, CA.

Gray, R. S., Kotz, D., Cybenko, G., & Rus, D. (1997). Agent Tcl. In W. Cockayne & M. Zyda (Eds.), *Mobile Agents*: Greenwich, CT: Manning Publishing.

Huber, M. J. (1999). JAM: A BDI-theoretic Mobile Agent Architecture. In *Proceedings of the Third International Conference on Autonomous Agents (Agents99)*, (pp. 236-243). Seattle, WA.

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.

John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interactions, 3*(4), 320 - 352.

Jones, R. (1998). A graphical user interface for human control of intelligent synthetic forces. In *Proceedings of the Seventh Conference on Computer Generated Forces and Behavioral Representation*, (pp. 631-635). Orlando, FL.

Jones, R.M. (1999). Graphical visualization of situational awareness and mental state for intelligent computer-generated forces. In *Proceedings of the Eighth Annual Conference on Computer Generated Forces and Behavioral Representation*. Orlando, Florida.

Jones, R.M. Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine.* Spring, 1999.

Kieras, D. E. (1998). *A guide to GOMS model usability evaluation using GOMSL and GLEAN3* (Technical Report No. 38, TR-98/ARPA-2). Ann Arbor: University of Michigan.

Kirwan, B., & Ainsworth, L. K. (1992). *A guide to task analysis*. London: Taylor & Francis.

Kumar, S., & Cohen, P. (2000). Towards a fault-tolerant multi-agent system architecture. In *Proceedings of the fourth international conference on autonomous agents*, (pp. 459-466). New York, NY: Association for Computing Machinery Press.

Kumar, S., Cohen, P., & McGee, D. (2001). Towards a formalism for conversation protocols using joint intention theory. *Computational Intelligence Journal (Special Issue on Agent Communication Languages), vol 18, no 2*. 174-228.

Labrou, Y. (1996). Semantics for an Agent Communication Language. Unpublished doctoral dissertation, University of Maryland Baltimore County, Baltimore.

Labrou, Y., & Finin, T. (1997). Semantics and conversations for an agent communication language. In Hunhs, M and Sign, M. (Eds.), *Readings in Agents*, New York, NY: Morgan Kaufmann,

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

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.

Leveson, N. G. (1995). *Safeware: system safety and computers*. Reading, MA: Addison-Wesley.

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

Maes, P. (1994). Agents that reduce work and information overload. *Communication of the ACM, 37*(7), 30-40.

Muller, J. P., & Pischel, M. (1994). Integrating agent interaction into a planner-reactor architecture. Paper presented at the *1994 Distributed AI Workshop*, Lake Quinalt, WA.

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

Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.

Peine, H. & Stolpmann, T. (1997). The architecture of the Ara platform for mobile agents. In Kurt Rothermel, Radu Popescu-Zeletin (Eds.): In *Proceedings of the First International Workshop on Mobile Agents MA'97* (Berlin, Germany), April 7-8th. Lecture Notes in Computer Science No. 1219, Springer Verlag, ISBN 3-540-62803-7.

Rosenbloom, P. S., Laird, J. E., & Newell, A. (Eds.). (1993). *The Soar Papers: Research in Integrated Intelligence*. Cambridge, MA: The MIT Press.

Rosenschein, J. (1985). *Rational Interaction: Cooperation among Intelligent Agents.*, Stanford University, Palo Alto, CA.

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.

Searle, J. (1970). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge, MA: Cambridge University Press.

Sheridan, T. B. (2000). Function allocation: Algorithm, alchemy, or apostasy? *International Journal of Human-Computer Studies, 52*, 203-216.

Shneiderman, B., & Maes, P. (1997). Direct manipulation vs. interface agents. *Interactions*, 42-61.

Shoham, Y. (1991). *AGENTO: A Simple Agent Language and Its Interpreter.* Paper presented at the Ninth National Conference on Artificial Intelligence, Anaheim, CA.

Shoham, Y. (1993). Agent-oriented Programming. *Artificial Intelligence, 60*(1), 51-92.

So, Y., & Durfee, E. H. (March, 1994). *Modeling and Designing Computational Organizations.* Paper presented at the Working Notes of the AAI Spring Symposium on computational Organization Design.

So, Y., & Durfee, E. H. (1997). Designing organizations for computational agents. In K. Carley, L. Gasser, & M. Prietula (Eds.), *Computational Organization Theory*: AAAI Press.

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

Thomas, R. S. (1995). The PLACA agent programming language. In M. Wooldridge & N. R. Jennings (Eds.), *Intelligent Agents - Theories, Architectures, and Languages,* (pp. 356-370): Springer.

Van Fosson, M. H. (2001). *Future Combat Systems*, [Presentation]. DARPA. Available: http://www.darpa.mil/DARPATech2000/Presentations/tto_pdf/3VanFossonFCSB&W.pdf [2001]

Wood, S. D. (1999). The application of GOMS to error-tolerant design. In *Proceedings of the 17th International System Safety Conference*, Orlando, FL.

Wooldridge, M. (2000). *Reasoning About Rational Agents*. Cambridge, MA: MIT Press.

Appendix A

Agent Message Types

Message Types

- Confirm
  - Sent By: Operator, Tasking Agent
  - Received By: Operator, Tasking Agent
  - Description: This type of message is either a request for confirmation, a confirmation, or a rejection of confirmation, depending on the Status parameter.

- Uninitialized
  - Sent By: Tasking Agent
  - Received By: Operator
  - Description: This type of message is a mission milestone status update to the operator.

- In-Progress
  - Sent By: Tasking Agent, Monitoring Agent
  - Received By: Operator, Tasking Agent, Coordinating Agent
  - Description: This type of message is a mission milestone status update.

- Failed
  - Sent By: Tasking Agent, Monitoring Agent
  - Received By: Operator, Tasking Agent, Coordinating Agent
  - Description: This type of message is a mission milestone status update.

- Complete
  - Sent By: Tasking Agent, Monitoring Agent
  - Received By: Operator, Tasking Agent, Coordinating Agent
  - Description: This type of message is a mission milestone status update.

- In-Jeopardy
  - Sent By: Tasking Agent, Coordinating Agent, Monitoring Agent
  - Received By: Operator, Tasking Agent, Coordinating Agent
  - Description: This type of message is a mission milestone status update.

- Request
  - Sent By: Tasking Agent, Coordinating Agent, Monitoring Agent
  - Received By: Operator, Tasking Agent, Coordinating Agent, Monitoring Agent
  - Description: This type of message is a request for a specific type of information from one agent type to another agent type or the operator.

- Report
  - Sent By: Tasking Agent, Coordinating Agent, Monitoring Agent
  - Received By: Tasking Agent, Coordinating Agent, Monitoring Agent

  - Description: This type of message is a report on a specific type of information from one agent type to another agent type or the operator.

- Move
  - Sent By: Operator, Tasking Agent
  - Received By: Entity
  - Description: This type of message is a movement order for an entity to a specific set of coordinates at a specified speed.

- Fire
  - Sent By: Tasking Agent
  - Received By: Entity
  - Description: This type of message is a direct or indirect fire order for an entity to a specific set of coordinates or target.

- Set-ROE
  - Sent By: Operator, Tasking Agent
  - Received By: Entity
  - Description: This type of message sets the Rules of Engagement for an entity.

- Halt
  - Sent By: Operator, Tasking Agent
  - Received By: Entity
  - Description: This type of message is a movement order for an entity to halt immediately.

- Initiate
  - Sent By: Operator
  - Received By: Tasking Agent
  - Description: This type of message is a command by the Operator to a Tasking Agent initiating a proposed plan.

- Update
  - Sent By: Operator
  - Received By: Tasking Agent
  - Description: This type of message is a command by the Operator to a Tasking Agent updating a plan already in the process of being executed.

- Terminate
  - Sent By: Operator
  - Received By: Tasking Agent
  - Description: This type of message is a command by the Operator to a Tasking Agent terminating a plan in the process of being executed.

Example Message Content

To: [Operator|Tasker|Coordinator|Monitor|<Entity Callsign>]
From: [Operator|Tasker|Coordinator|Monitor|<Entity Callsign>]
Subject: [<Message Type>]
<Data Member Attribute1>: [<Data Member Attribute-Value1>]
<Data Member Attribute2>: [<Data Member Attribute-Value2>]

Example Message

To: Tasker
From: Operator
Subject: Confirm-Fire-Order
Status: Confirmed

Appendix B

Agent Communications UML Sequence Diagram

