I-RESCUE: A COALITION BASED SYSTEM TO SUPPORT DISASTER RELIEF OPERATIONS

Clauirton Siebra and Austin Tate
Artificial Intelligence Applications Institute
Centre for Intelligent Systems and their Applications
School of Informatics, The University of Edinburgh
Appleton Tower, Crichton Street, EH9 9LE, Edinburgh, UK
{c.siebra,a.tate}@ed.ac.uk

Abstract

I-Rescue is a research programme that aims to develop knowledge-based tools for disaster relief domains. One important aspect of the I-Rescue development is to highlight the requirements regarding the collaborative activities of planning and execution, considering a hierarchical structure of decision-making and a mixed-initiative style of interaction between users and systems. This paper discusses the design and implementation of I-Rescue and its use in a search and rescue domain where joint users, assisted by customised agents, are able to perform complementary tasks.

Key Words

Coalitions, multiagent planning, disaster relief domains.

1. Introduction

Coalition support systems (e.g., [1]) are applications that integrate different human and/or software agents so that they are able to work together to achieve mutual objectives. An important feature in systems like that is their ability to support the collaborative activities of planning and execution. During planning processes, joint agents share knowledge so that a plan can be built in accordance with the perspectives of each agent. Then the activities in that plan are assigned to specific agents, which will use their individual capabilities to perform the allocated tasks. As real domains are generally dynamic, agents are always interleaving planning and execution [2]. so that they can adapt themselves to new conditions.

Several works in AI have proposed frameworks to collaborative systems. SharedPlans [3] provides a specification for design of collaborative-capable agents that considers the interleaving of planning and execution, commitments that can lead the agents' behaviour and constraints that avoid conflicting intentions. STEAM [4] is an implemented model of teamworks, based on Joint Intention Theory [5], that enables explicit representation of team goals and plans, and teams' joint commitments. The work of Kinny [6] considers the use of pre-planned team activities as a way of collaborative agents to respond rapidly to important events by adopting a specific plan from their repertoire. O-Plan [7,8] provides an architecture within which different agents have command (task assignment), planning and execution monitoring roles.

Although these (and others) works have different approaches to deal with different technical problems, they agree that agents involved in collaborative environments need to make commitments on joint activities, reach consensus on plans and also make commitments to the constituent activities of such plans.

As in these works, we are investigating ways of describing collaborative activities, in particular, for coalition support systems. Differently from standard collaborative systems, as described in Grosz [9], coalitions are usually not composed of a teams of equals, but present a hierarchical structure of command and control [10] where agents take different (and complementary) kinds of decisions at each level.

Another important factor is that decisions are taken in a mixed-initiative style, where computational agents assist human users. Consequently, users and system agents will play different roles inside the decision processes. While users have the ability to take decisions based on their past-experiences (case-base reasoning), system agents are able to generate and compare a significant number of options, showing both positive and negative points of such options.

Our ongoing framework for coalition systems' development intends to consider explicitly these features inside collaborative descriptions, so that those notions are easily mapped to a practical application. In this way, the principal aim of this work is to design and implement a coalition support system that can be applied to practical real domains, in particular search and rescue scenarios, so that via its development process we can obtain the requirements to support joint activities.

The remainder of this document is structured as follows: section 2 introduces the I-Rescue architecture and its levels of decision. Section 3 presents the I-Rescue design process, highlighting its components' models. Section 4 describes a practical application of I-Rescue in a disaster relief domain. Section 5 discusses some conclusion and future works, while the last part lists significant references.

2. I-Rescue Architecture

I-Rescue identifies three levels of decision-making (strategic, tactical and operational) in which different kinds of tasks are performed by the agents (Fig. 1).

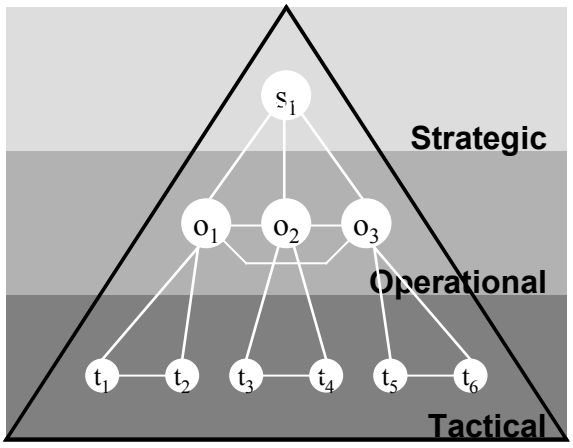

Fig.1: I-Rescue architecture and their three levels of decision

Strategic agents (analysis and directions) build plans in a high-level of granularity ("what-to-do plan"). They deal with diversified and no-technical information, and their issues are normally accomplished in long terms. Basically whole coalition is affected by its decisions;

Operational agents (synthesis and control), in general, account for refining the plans produced in the strategic level, deciding who will carry out the tasks ("who-to-do plans") via processes such as resource scheduling and load balancing. Thus information is more specialised and limited groups of agents will be affected by their resolutions.

Tactical agents (reaction and execution) are the agents that, in fact, accomplish the tasks ("how-to-do"). For this reason, their level of knowledge is well specialised on the domain that they are operating and their decisions are taken on sets of atomic actions.

This hierarchical arrangement is a common practice in military models of command and control [11], and consistent with knowledge engineering work[1]. We can note that the different features of each level influence the specification of mechanisms that the system can provide for it. Thus I-Rescue, following the IP^2 approach (section 4.2), enables agents to be customized via an open plug-in interface, so that capabilities can be added or changed in accordance with new requirements.

Each agent (s_1, o_1, o_2, etc.) in this structure (Fig. 1) is, in fact, a set of human user(s) and computational agent that interact in a mixed initiative style. This approach results in several advantages: intensifies the user control and involvement, permits user interaction during the whole decision process so that users are able to understand why ways were chosen or avoided, and removes the premise of complete and bug-free knowledge.

In addition to assisting human users during cognitive processes, the computational agents also account for communication functions. In order each white line (Fig. 1) represents both a communication channel and a relation pattern. Contrasting to this simple figure, the agents do not necessarily have a peer-to peer communication, but they can make use of a facilitator component that realises the tasks of agents' registering and messages' routing.

Relation patterns define the kind of relationship between two agents. Basically there are two kinds of relation pattern: peer-peer and superior-subordinate. The first pattern can occur between agents of the same level (e.g. t1 and t2). The second can happen between agents of different (and adjacent) levels. Policies can be defined on relation patterns as a way of restricting the kind of communication that two agents can maintain during an operation.

3. Design and Components

A mutual understanding of the coalition components, based on a shared underlying conceptual model, can provide a framework for systems to support the collaborative processes of planning and execution. Next sub-sections discuss how an ontology (in our case the <I-N-C-A> ontology [12]) can provide this mutual understanding, and which components can be specified via that ontology.

3.1 <I-N-C-A> Ontology

Ontologies are formal descriptions of concepts and relationships that can exist for organisations and their agents. Ontologies are designed for the purpose of enabling knowledge sharing and reuse [13] via agreements to use a vocabulary in a way that is consistent with the theory pre-specified. Considering this aspect of ontologies, the I-Rescue design makes use of <I-N-C-A> ontology to underpin the collaborative planning and execution processes.

[1] KADS methodology, for example, separate the different "task types" (diagnosis, interpretation, monitoring, etc.) into three classes: analysis, synthesis and modification tasks.

<I-N-C-A> (Issues – Nodes – Constraints - Annotations) can be used to represent a plan as a set of constraints on the space of all possible options in the application domain. Each plan is considered to be made up of a set of "Issues" and "Nodes". Issues represent potential requirements that need to be considered at sometime. Nodes represent activities in the planning process that may have parts called sub-nodes making up a hierarchical description of plans. Nodes are related by a set of detailed "Constraints" of diverse kinds such as temporal, sequential, priority and so on. "Annotations" add complementary human-centric and rationale information to the plan.

We can use these definitions to specify the components that have influence on the plan building. The next sub-section discusses our approach to carry out that task.

3.2 Components of a Coalition System

The design process for a coalition support system, such as I-Rescue, can be anchored in a component model as in the graph below (Fig. 2). According to the graph, a whole coalition system can be modelled by the specification of their agents, domain objects, processes and relations between such elements.

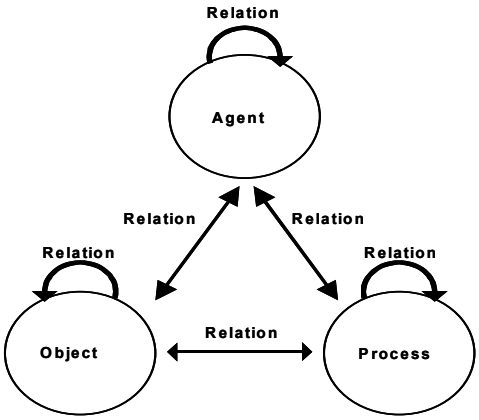

Fig. 2: Abstract model for coalition support systems design

Agents can be described by their capabilities in solving specific problems and by the constraints on such capabilities. That information is important to the planning process because it supports the tasks of localizing agents that are able to perform specific activities and also restricts the options during the plan building. In a collaborative planning environment, the system can use agents' descriptions to compound new capabilities through the interoperation and union of others existent capabilities so that complex tasks can be solved.

Agent-agent relations describe (via policies on these relations) the kind of interaction that agents can maintain inside the organisation. A policy can specify, for example, that an agent cannot delivery tasks to another agent in a peer-peer relation. One of the principal functions of this kind of relation is to organise the hierarchical structure of the system, specifying the levels of decision-making and, indirectly, the groups that account for a specific kind of problem.

Objects are the individual components that form the domain. As I-Rescue is designed to support operations in disaster relief domains, their objects are represented, for example, for buildings in fire, blocked roads and so on. Each object is specified via a set of pattern-values pairs, which is a very simple and general representation to be used by the planning system.

Object-object relations describe links between individual objects. For example, roads of a city can be connected to form a graph so that the system can apply mechanisms like a pathfinder or regions' patrolling. That relation is also specified via sets of pattern-values pairs, where patterns have a special semantic such as "partOf", "linkedTo" or "unionOf".

Processes are specified as set of activities (nodes), which are carried out by different agents of the coalition. Processes also have a set of constraints that guide their planning and execution, together with associated agents' and objects' constraints.

Process-process relations specify constraints that restrict the merging of different processes, showing if they can be performed in parallel (or sequentially), and which conditions the processes need to provide to enable others.

Agent-object relations capture the "desires" of agents regarding specific states of objects. This aspect is described via issues. **Agent-process** relations describe responsibilities of joint agents regarding processes. Finally **process-object** relations state the status of changing that processes are realising on objects, so that users can monitor the evolution of plans.

4 Disaster Relief Application

This section discusses the implementation and use of I-Rescue in a disaster relief domain. To that end, we are using a fictional disaster scenario based on the Japanese city of Nagoya. The next sub-sections discuss the motivation and issues for such domain, the tools to present and manipulate the components' models and how the system uses that information during the collaborative planning and execution processes.

4.1 Nagoya Domain

Japan is a prone region to earthquakes due to the number of tectonic plates that converge below the country's surface. The Kobe Earthquake of January 1995 is an

example of how disasters like that have tragic effects in urbane areas. To lighten the damage of earthquakes in the future, scientists are studying ways to predict the occurrence of quakes more accurately. One of the results of that study was to appoint the Chubu region (Fig. 3) to be a candidate for a large quake in the near future.

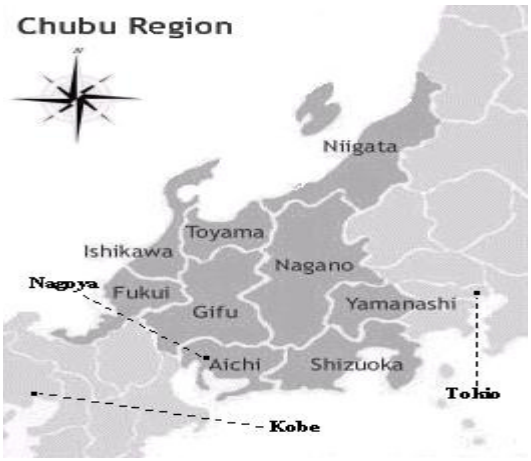

Fig.3: Chubu Region (Centre of Japan), showing the location of Nagoya

Several events can happen during a large earthquake: buildings collapse, civilians are buried under the collapsed buildings, fires spread through the city, roads are blocked turning complex the evacuation of civilians and the work of rescue teams. We can note that scenarios like that require different kinds of abilities such as medical teams to take care of injured people, fire brigades to extinguish fires, search teams to look for buried people, police to control the evacuation of the city and so on.

In this way, I-Rescue has the objective of assisting each of these agents, organising them into a hierarchical structure of command and control. Via computational agents, represented by I-X Process Panels (section 4.2), members of a coalition will be able to participate during the processes of planning and execution, sharing knowledge and optimising the use of resources.

4.2 Tools and Practical Requirements

I-X Process Panels (IP^2) [14] involves a set of concepts and tools that implement the design ideas discussed in section 3. To explain such implementation, we use a subset of I-Rescue agents, which are summarised in the table below (Tab. 1).

Agent	*Level*	*Main activity*
SRC*	Strategic	Build high-level plan
NagoyaHosp.	Operational	Coordinate ambulances
Ambulance1	Tactical	Rescue injuries
Ambulance2	Tactical	Rescue injuries

Tab. 1: Subset of I-Rescue agents, showing their level of operation and main activity. *(SRC = Search and Rescue Centre)

I-Rescue agents are organised via the *I-Space* tool (Fig. 4), which manages hierarchical relationships of a specific agent (e.g. Nagoya Hospital) to other agents and external services (e.g. the *bloodBank* is a service that informs the available amount and kind of blood to hospital). Particular actions can be associated to each of the relations (e.g. "delegate to" action is only possible to subordinate agents).

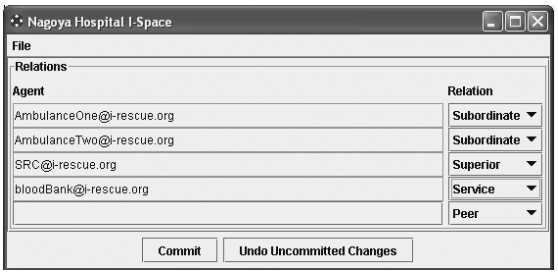

Fig. 4: I-Space tool and their possible relations (subordinate, superior, service and peer)

The system codifies each of these agents as a set of pattern-value pairs (constraints). An IP^2 pattern is, in general, an n-tuple, currently though we are using two elements in the patterns: the first to keep the attribute and the second to keep the object ID. So we can represent all subknowledge as (<attribute> <object>= <value>). For example, the *AmbulanceOne* agent could be described as:

(agentType AmbulanceOne) = Ambulance
(latitude AmbulanceOne) = 35.5674
(longitude AmbulanceOne) = 136.5202
(maxSpeed AmbulanceOne) = 120km/h
(FuelTank AmbulanceOne) = 43%

The environment is described as facts in the same format, and the system deals with both agents and environment features as states of world. S*tate viewers* (Fig. 5) account for displaying the current world state of an agent (its beliefs) and it can be implemented to present the information in different ways and perspectives. In our case, for example, we are using a map tool as way to capture the space-time relation among objects.

Modifications in the states of objects can trigger off new issues. For example the low level of blood stocked in the hospital will produce the new issue of "refilling blood stock". This issue has a priority (lowest, low, normal, high or highest) and it needs to be considered by the user at sometime. Agents can create issues by themselves, or issues will be the result of income messages from superior agents.

Issues can be transformed in activities when users are deliberating on them. Both concepts have a similar structure, presenting priorities and sets of associated constraints. However issues are dealt as questions to be handled and can represent "desires" that are waiting for a proper moment to be considered. Thus they do not consume resources.

On the other hand activities consume resources (time, physical devices, energy, etc.) and agents have a set of mechanisms to deal with them. A simple way of dealing with an activity is to delegate to another agent that has a better ability to perform it. However mechanisms can be implemented in accordance with the tasks that are carried out on each level. For example, tactical agents (such as ambulances) that have the task of rescuing an injured civilian could apply a pathfinder mechanism to find the best route to such civilian. As the pathfinder mechanisms is implemented in accordance with the plug-in interface of I-Rescue, it is able to access world state instantiations and return a route that avoids, for example, the blocked roads.

All interactions and knowledge sharing between agents (IP^2) are supported by a communication strategy. Currently I-Rescue uses Jabber Technology [15] due to its support to XML format, instant messages and presence concepts. However the modular IP^2 implementation allows the use of others strategies as well.

4.3 Planning and Execution

Normally the planning process starts on the strategic level, where users have a global vision of the environmental situation. To handle its income issues, the strategic agent decides on a set of activities to be assigned to its subordinate agents. This process can be carried out in two different ways. First users can make use of pre-planned libraries and choose sequences of activities that are able to handle their issues. SOPs (Standard Operation Procedures) [16] are examples of pre-defined activities, based on experiences, lessons learned or careful pre-design, which can be used in this process.

The other option is to call an intelligent planning system to find a suitable set of activities. As the strategic level does not have detailed information about the duration of these activities, it can use a qualitative model of time (e.g. $task_1$ *after* $task_2$) to synchronize them. The planning process can be seen as a constraint satisfaction problem, where both environment and subordinate agents impose the set of constraints. The constraints imposed by agents represent the perspectives and knowledge of such agents, so that the constraint assignment is the way that they influence the planning process in a mixed-initiative fashion.

Operational and tactical levels act in a similar way by adding further constraints into the plan. The difference is the kind of decision made for each of them and the mechanisms that they use to do that.

While the use of a qualitative time model by the strategic level is necessary to avoid it taking decisions about things that it does not care about (using the least commitment principle), the operational level can use a quantitative model of time (e.g. $task_1$ *duration* 29min) to improve decisions. A quantitative model of time enables resource allocation processes to increase the amount of concurrent acting, avoiding agents having to wait a long time to be allocated tasks. As agents have no control on the duration of some tasks in disaster domains (e.g. extinguishing fires, finding buried people, etc.), duration values are usually estimations that can be based on past-cases and the capacities of performing agents. In this way, each operational agent needs to have an appropriate description of its subordinate agents.

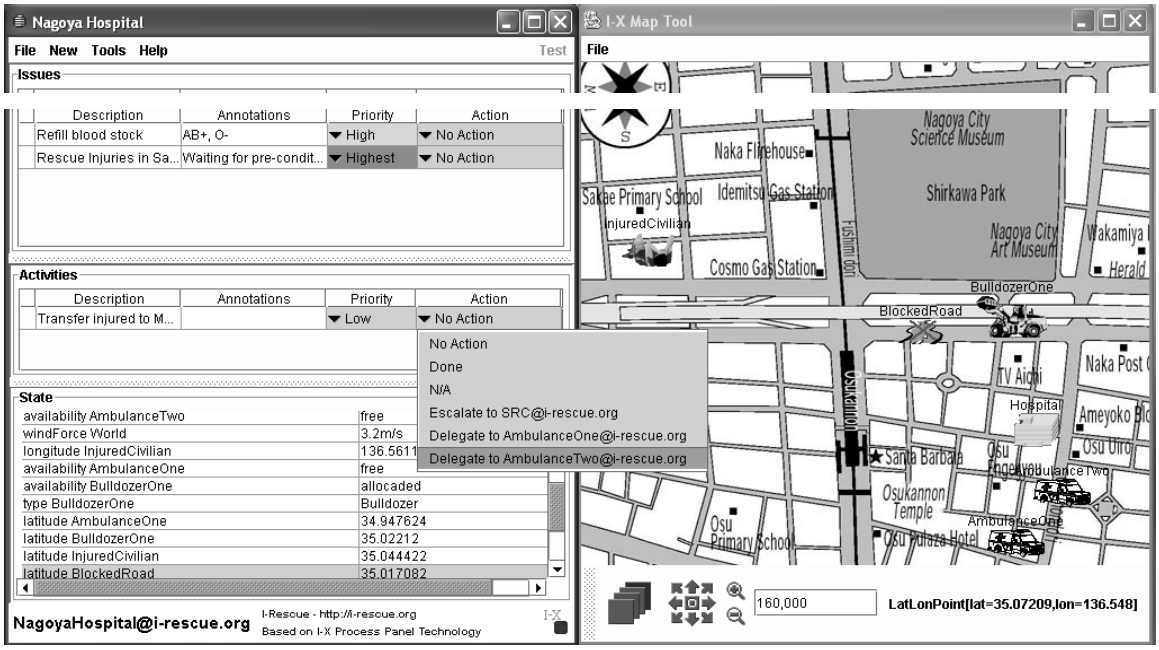

Fig. 5: I-Rescue Agent to Nagoya Hospital, highlighting its issues, activities, and state viewers (table and map tool)

The planning processes in the tactical level are frequently reactive, consisting of pre-defined sensor-activity rules. In this way, agents are always adjusting their planned activities as they gather new information about the environment or encounter unexpected events. However these adjustments (short-term decisions) need to be in accordance with the commitments already made. Agents at this level need to follow their commitments because they are not able to evaluate how their decisions affect the coalition's long-term goals. If an agent cannot deal with their tasks by itself, it needs to report this fact to the upper level (operational level).

The concept of "reports" is very important into these processes. Via reports the system can follow the ongoing activities, detecting possible failures in the plan. Progress and completion reports are associated with activities and agents are encouraged to send them in accordance with the evolution of their activities.

5. Conclusion and Future Works

The I-Rescue design is intended to support the collaborative processes of planning and execution. It uses a structure of issues, activities and state as form of organizing the knowledge and processes of human users and system components so that they are each able to understand their collaborative role inside the coalition.

The requirements of practical tools to manipulate the internal models of I-Rescue (relationships, processes, agents, etc.) were investigated during its implementation, considering a potential application in search and rescue domain. This set of practical requirements will be considered together with elements of related works in multiagent mixed-initiative collaboration research so that a framework to specify coalition systems can be produced from perspective of the software engineer, but without losing the formalism of logical descriptions.

6. Acknowledgement

Clauirton Siebra' scholarship is sponsored by CAPES Foundation under processes number BEX2092/00-0. This material is based on research within the I-X project sponsored by the Defense Advanced Research Projects Agency (DARPA) and US Air Force Research Laboratory under agreement number F30602-03-2-0014 and others sources.

The University of Edinburgh and research sponsors are authorised to reproduce and distribute reprints and on-line copies for their purposes not withstanding any copyright annotation here on. The views and conclusions contained here in are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of other parties.

References

[1] D. Allsopp, P. Beautement, J. Bradshaw, E. Durfee, M. Kirton, C. Knoblock, N. Suri, A. Tate and C. Thompson, Coalition Agents Experiment: Multiagent Cooperation in International Coalitions, *IEEE Intelligent Systems, 17*(3), 2002, 26-35.

[2] J. Ambros-Ingerson, *IPEM: Integrating Planning, Execution, and Monitoring*, PhD thesis, Department of Computer Science, University of Essex, 1987.

[3] B. Grosz, L. Hunsberger and S. Kraus, Planning and Acting Together, *AI Magazine, 20*(4), 1999, 23-34.

[4] M. Tambe, Towards Flexible Teamwork, *Journal of Artificial Intelligence Research, 7*, 1997, 83-124.

[5] J. Levesque, P. Cohen and J. Nunes, On Acting Together, *Proc. AAAI National Conference on Artificial Intelligence*, Menlo Park, CA, 1990.

[6] D. Kinny, M. Ljungberg, A. Rao, G. Tidhar and E. Werner, Planned Team Activity, *Proc. 4th European Workshop on Modelling Autonomous Agents in a Mulit-Agent World*, Rome, Italy, 1992.

[7] A. Tate, Coordinating the Activities of a Planner and an Execution Agent, *Proc. 2nd NASA Conference on Space Telerobotics*, Chicago, USA, 1989, 385-393.

[8] A. Tate, J. Dalton and J. Levine, O-Plan: a Web-based AI Planning Agent, *Proc. AAAI Intelligent Systems Demonstrator*, Austin, Texas, USA, 2000.

[9] B. Grosz, Collaborative Systems, *AI Magazine, 17*(2), 1996, 67-85.

[10] N. Heacox, M. Quinn, R. Smillie, J. Hayes and J. Jensen, Evolution of a Mission Plan, *Proc. 6th International Command and Control Research and Technology Symposium*, U.S. Naval Academy, Annapolis, Maryland, USA, 2001.

[11] T. Killion, Decision Making and the Levels of War, *Military Review, 80*(6), 2000, 66-70.

[12] A. Tate, <I-N-C-A>: an Ontology for Mixed-Initiative Synthesis Tasks, *Proc. IJCAI Workshop on Mixed-Initiative Intelligent Systems*, Acapulco, Mexico, 2003.

[13] T. Gruber, Toward Principles for the Design of Ontologies Used for Knowledge Sharing, *Journal of Human-Computer Studies, 43*(5/6), 1995, 907-928.

[14] A. Tate, J. Dalton and J. Stader, IP^2 – Intelligent Process Panels to Support Coalition Operations, *Proc. 2nd International Conference on Knowledge Systems for Coalition Operations*, Toulouse, France, 2002, 184-190.

[15] P. Saint-Andre, Jabber Technology Overview. *Jabber Software Foundation*, 2001.

[16] US Joint Publication for Joint Tactics, Techniques, and Procedures for Combat Search and Rescue, *Joint Publication 3-50.21*, Joint Chiefs of Staff, 23 March 1998.