Rapid Integration and Coordination of Heterogeneous, Distributed Agents for Collaborative Enterprises

David V. Pynadath, Milind Tambe, Nicolas Chauvat
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{pynadath,tambe,nico}@isi.edu

Abstract

As the agent methodology proves more and more useful in organizational enterprises, research/industrial groups are developing autonomous, heterogeneous agents that are distributed over a variety of platforms and environments. Rapid integration of such distributed, heterogeneous agent components could address large-scale problems of interest in these enterprises. Unfortunately, rapid and robust integration remains a difficult challenge. To address this challenge, we are developing a novel teamwork-based agent integration framework. In this framework, software developers specify an agent organization through a team-oriented program. To locate and recruit agent components for this organization, an agent resources manager (an analogue of a "human resources manager") searches for agents of interest to this organization and monitors their performance over time. TEAMCORE wrappers render the agent components in this organization team ready, thus ensuring robust, flexible teamwork among the members of the newly formed organization. This implemented framework promises to reduce the development effort in enterprise integration while providing robustness due to its teamwork-based foundations. We have applied this framework to a concrete, running example, using heterogeneous, distributed agents in a problem setting comparable to many collaborative enterprises.

1. Introduction

An increasing number of collaborative enterprises are turning to agent technology to address the complex, dynamic environments common to most enterprises [1, 4]. As more agents populate our organizations, whether in businesses, the military, etc., there is a more critical need to rapidly marshal the agents needed for newly arising tasks and to enforce proper coordination among these agents. Unfortunately, although the agent methodology does provide a great simplification over directly integrating legacy systems, the problem of coordinating agent behavior in a large-scale system remains difficult.

First, in the distributed, open environments found in most enterprises, there are great difficulties in identifying and accessing all of the agents relevant to the particular information and control needs of a new task. Second, since the recruited agents are not usually built to work together, building an integrated system with appropriate coordination among the agent components is difficult. Third, the resulting integrated system must be robust, in that it ensures the desired output despite the uncertainties of a dynamic, open environment. There are other issues in integration as well (e.g., common communication language), but this short note focuses on only the three key challenges mentioned here.

To address these integration challenges, our TEAMCORE project focuses on enabling enterprise managers to build large-scale agent organizations. There are currently two key aspects to this project. The first focuses on the creation, specification, and monitoring of the agent organization. The second focuses on enabling the organization to reliably execute tasks, by ensuring robust teamwork among the agents in the organization.

KARMA (*Knowledgeable Agent Resources Manager Assistant*) addresses the first aspect by assisting enterprise managers in three ways. First, Karma aids in *team-oriented programming*, where the system designer specifies a hierarchical agent organization, as well as its high-level goals (e.g., plan hierarchy, supply chain). Team-oriented programming abstracts away from coordination details, thus eliminating the burden of writing large numbers of coordination plans. Second, Karma locates agents that match the specified organization's requirements and assists in allocating organizational roles to such agents, thus alleviating the burden of searching through the vast numbers of agent components present in most enterprises. Third, Karma monitors the organization to diagnose failures and to evaluate agent performance for future (re)organizations. Karma's agent resources management functionality differs significantly from *middle agents* such as matchmakers. If middle agents are the analogues of the "middle-men" of physical commerce [2], then agent resource managers are the analogues of the "human resources managers" of a commercial firm. Agents such as Karma will become increasingly critical with the increasing number of agent components available within an enterprise (and eventually across multiple enterprises).

Once the enterprise manager has specified a team-oriented program, the second aspect of the TEAMCORE project focuses on robust execution. Agent teamwork enhances robust execution, since we can expect agent components, as team members, to act responsibly towards one another, covering for each other's execution failures and sharing key information. To enable such teamwork among independently designed agent components, we make the agents *team-ready* by providing each with a separate TEAMCORE wrapper. We thus avoid the need to modify the agents themselves, an important consideration when the agent components can be complex legacy systems. The TEAMCORE wrappers are based on STEAM, a reusable, general-purpose teamwork module that encapsulates reasoning about common teamwork coordination, including contingencies in such coordination [6]. Given this model, the TEAMCORE wrappers automatically generate the required coordination actions in executing a team-oriented program. Thus, TEAMCORE shields the human developer from the responsibility of designing all such specifications.

We have applied our framework to a concrete problem from a military domain that shares many features with most enterprise integration tasks. In our example problem, we successfully integrated various information and control agents to provide a simulated mission rehearsal of the evacuation of civilians stranded in a hostile area. The integrated system had to enable a human commander to interactively provide locations of the stranded civilians, safe areas for evacuation, and other mission parameters, and to then have simulated helicopters fly a coordinated mission to evacuate the civilians. The integrated system plans routes to avoid known obstacles, obtains information about dynamic enemy threats, and changes routes when needed. Thus, we have an information and task dependency structure similar to the supply chains found in many collaborative enterprises.

Our framework enabled the construction of such an integrated system out of 11 different existing components, including a multi-modal user interface agent, a route-planner, a web-querying information-gathering agent and synthetic helicopter pilots. Different developers, using four different computer languages, designed these agents, which ran on two different operating systems, on machines distributed across the United States. The heterogeneity and decentralization present in this military enterprise reflect the reality of most collaborative enterprises. Here, we used Karma to specify the necessary team-oriented program and to successfully locate relevant agents through an interface with a matchmaker and other middle agents. The chosen agents, with their TEAMCORE wrappers, successfully executed the team-oriented program, with this execution being robust against agent failures and other dynamic events similar to those that arise in most real-world enterprises.

2. Karma & TEAMCORE

Figure 1 illustrates how one builds agent organizations through the Karma-TEAMCORE framework. The numbered arrows show the typical stages of this process. In stage 1, human enterprise managers use TOPI (Team-Oriented Programming Interface) to specify a team-oriented program, with an organization, its goals, and its plans. TOPI in turn passes on the program specification to Karma (stage 2). In stage 3, Karma derives the requirements for roles in the organization and searches for agents with relevant expertise (labeled "domain agents" in Figure 1). Karma can query different middle agents, an Agent Naming Service (ANS) white pages, and other directory services. Karma then aids the developer in assigning these agents to organizational roles.

Having thus fully defined a team-oriented program, Karma launches the TEAMCOREs. Each TEAMCORE wraps an individual domain agent assigned to the organization, and the teams of wrappers jointly execute the team-oriented program. While executing this program, the TEAMCOREs broadcast information among themselves via multiple broadcast nets (stage 4). TEAMCOREs also communicate with the domain agents (stage 5). Karma "eavesdrops" on the various broadcasts to monitor the progress of the teams (stage 6), and it displays this progress to the enterprise manager.

A key novelty and strength of our framework is that powerful teamwork capabilities are built into its foundations, i.e., in the TEAMCORE wrappers. These wrappers enable agents, not originally constructed as cooperative, to still plan and act together as a team. Thus, in a team formed with TEAMCORE wrappers, agents automatically cover for failed teammates, supply key information to each other, etc. This framework strongly contrasts with previous agent integration frameworks such as the Open Agent Architecture (OAA) [5], where centralized facilitators enable agents to locate each other, but do not provide teamwork capabilities. In addition, TEAMCORE's distributed approach avoids a centralized processing bottleneck, as well as eliminating any central point of failure.

3. Karma: Specifying and Monitoring Team Programs

Karma helps a enterprise manager in three tasks important in the building of an agent organization: (i) specifying a team program; (ii) locating and assigning relevant agents; (iii) monitoring and recording agent performance.

The Team-Oriented Program A enterprise manager specifies an organization of interest via a team program: (i)

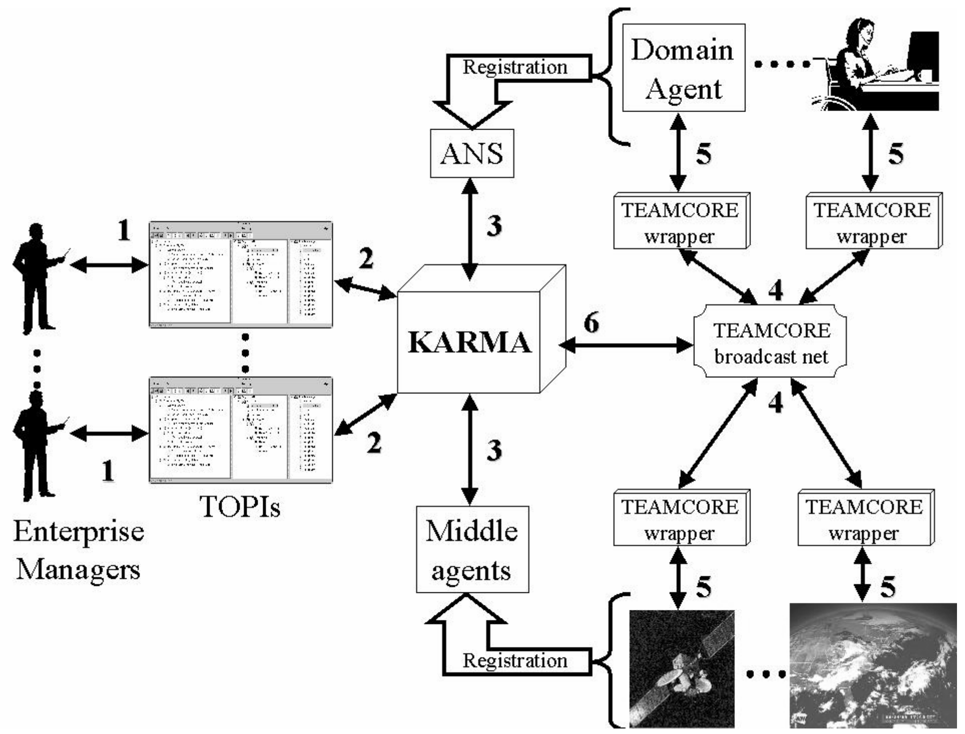


Figure 1: The overall Karma-TEAMCORE framework. TEAMCOREs may wrap different "domain agents" which may include information gathering agents, user assistants etc.

an organization hierarchy; (ii) a plan hierarchy; and (iii) assignments of agents to execute plans. The team organization hierarchy consists of roles for individuals and for groups of agents. For example, Figure 2-a illustrates a portion of the organization hierarchy involved with the evacuation scenario. Each leaf node corresponds to a role for an individual agent, while the internal nodes correspond to teams of these roles. *Task Force* is thus the highest level team in this organization, while *Orders-Obtainer* is an individual role.

The second aspect of a team program is a hierarchy of team plans explicitly expressing the joint activities of the relevant team. These plans describe the appropriate actions, the agents performing these actions, and additional high-level coordination knowledge (e.g., relevant information to be shared). The generality of the plan representation could potentially represent multi-level supply chains as the joint activity of suppliers and consumers. Figure 2-b shows an example from the evacuation scenario (please ignore the bracketed names for now). Here, high-level team plans, such as **Evacuate**, typically decompose into other team plans, such as **Process-orders**, to interpret orders provided by a human commander.

The third aspect of team-oriented programming is the assignments of agents to plans. The developer first assigns the organization's roles to plans and then assigns agents to these roles. Assigning only abstract roles rather than actual agents to plans provides a useful level of abstraction, since we can more quickly (re)assign new agents more quickly as needed. Figure 2-b shows the assignment of roles (in brackets) to the plan hierarchy for the evacuation domain. Associated with each plan is a specification of the requirements to perform the plan. A role inherits the requirements from each plan that it is assigned to.

The team program offers the key advantage of omitting the details of how to realize the specified coordination. Thus, for instance, the team designer does not program any synchronization actions — instead, during execution, the TEAMCORE wrappers automatically enforce the correct synchronization actions, both with respect to the time of plan initiation, the choice of plan, and the time of plan termination.

To facilitate the encoding of the team-oriented program, Karma interacts with a developer via the TOPI interface. Figure 3 shows a sample screenshot from programming the evacuation scenario, where the three panes correspond to the plan hierarchy (left pane), organization hierarchy (middle pane), and the domain agents (right pane). The left pane reflects the diagram 2-b. Associated with each entity are its properties, including its coordination constraints, preconditions, assigned subteam, etc.

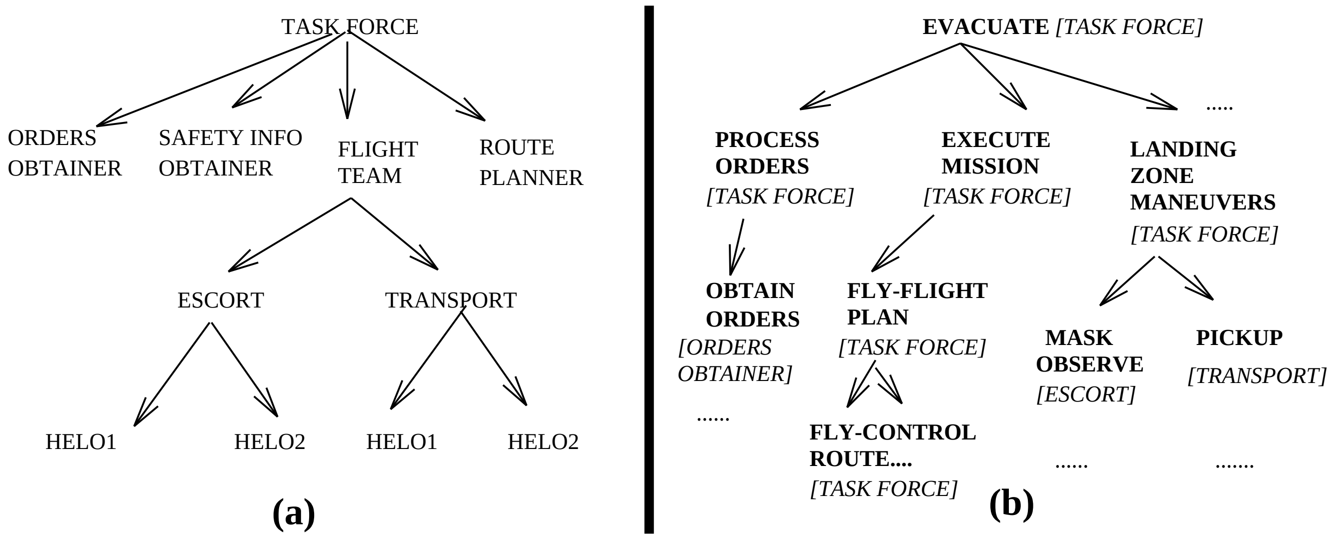
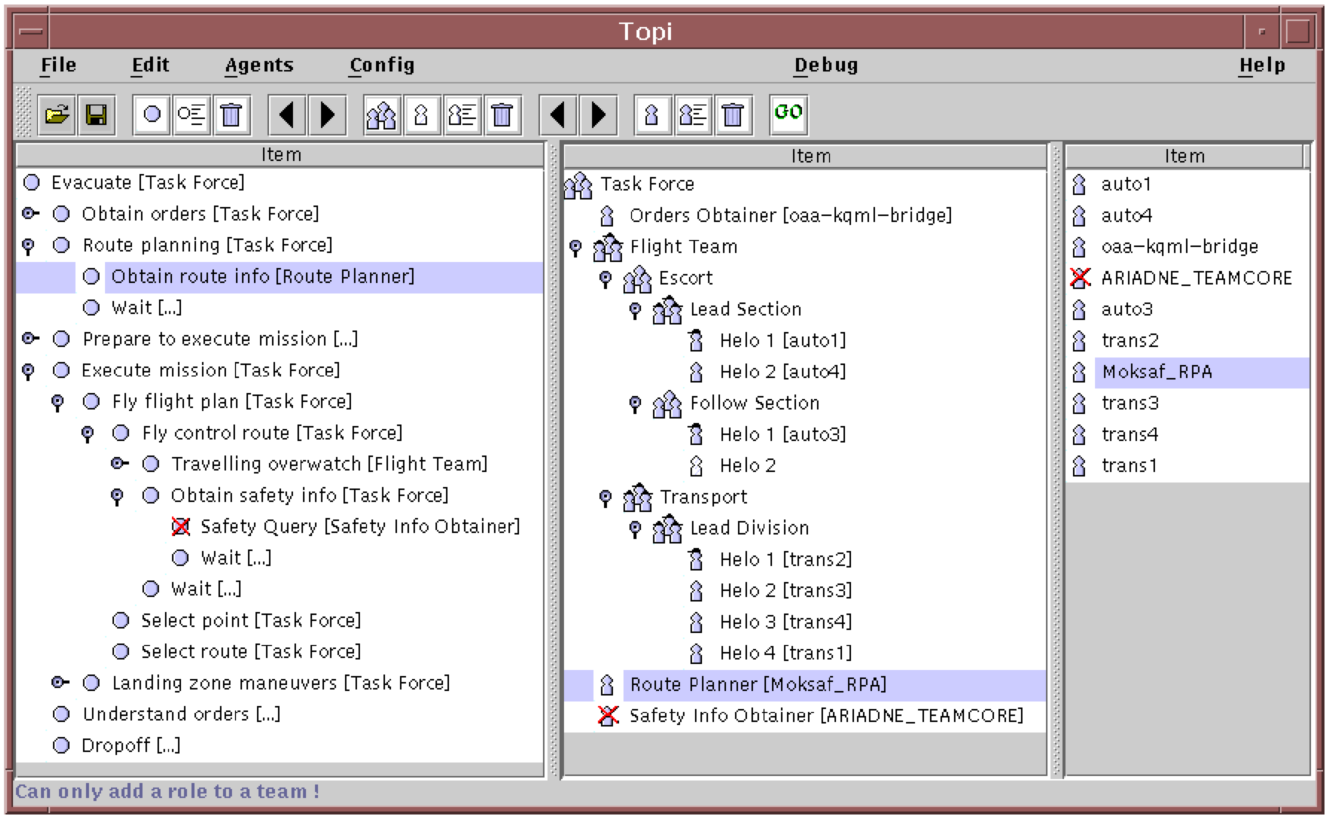


Figure 2: (a) Partial organization hierarchy with roles; (b) Partial reactive team plan hierarchy, both for the evacuation scenario.



```
Evacuate [Task Force]
  Obtain orders [Task Force]
  Route planning [Task Force]
    Obtain route info [Route Planner]
    Wait [...]
  Prepare to execute mission [...]
  Execute mission [Task Force]
    Fly flight plan [Task Force]
      Fly control route [Task Force]
        Travelling overwatch [Flight Team]
        Obtain safety info [Task Force]
          Safety Query [Safety Info Obtainer]
          Wait [...]
        Wait [...]
      Select point [Task Force]
      Select route [Task Force]
    Landing zone maneuvers [Task Force]
  Understand orders [...]
  Dropoff [...]
```

```
Task Force
  Orders Obtainer [oaa-kqml-bridge]
  Flight Team
    Escort
      Lead Section
        Helo 1 [auto1]
        Helo 2 [auto4]
      Follow Section
        Helo 1 [auto3]
        Helo 2
    Transport
      Lead Division
        Helo 1 [trans2]
        Helo 2 [trans3]
        Helo 3 [trans4]
        Helo 4 [trans1]
  Route Planner [Moksaf_RPA]
  Safety Info Obtainer [ARIADNE_TEAMCORE]
```

```
auto1
auto4
oaa-kqml-bridge
ARIADNE_TEAMCORE
auto3
trans2
Moksaf_RPA
trans3
trans4
trans1
```

Can only add a role to a team !

Figure 3: TOPI: The team-Oriented programming interface.

Searching and Assigning Agents Once Karma derives requirements for individual roles based on their assigned plans, it searches for agents with matching capabilities. Karma searches multiple sources: middle agents, local white pages directories of known agents, and other registry services. For instance, Karma may query the AMatchMaker [2] middle agent by sending it a KQML message specifying an advertisement template. AMatchMaker returns descriptions of those agents whose advertised capabilities match the template. In addition, Karma can search its own database of previously used agents. More recently, we have interfaced Karma with "the Grid" (produced by DARPA's COABS program [3]), an agent infrastructure providing registration and other interconnection services.

Karma compiles a list of relevant agents from these different sources, specifying their properties, including communication address, capabilities, etc. From this list, the developer can assign agents to the roles in the specified organization. By limiting its search to just those agents that meet the organizational requirements, Karma avoids overwhelming the system designer with unnecessary information.

Monitoring and Recording Agent Performance While the team executes its program, Karma's task shifts to monitoring and recording the execution. Currently, Karma's observations trigger feedback in TOPI, showing the developer which team plans and domain agents are currently active. To facilitate the reuse of agents across multiple tasks (or multiple runs of the same task), Karma also records how well each agent performs in the current task. Thus, upon completion of the task, each TEAMCORE wrapper sends Karma a report regarding the wrapped agent's performance, including any catastrophic failures during a run, success during runs, response times etc. Karma records this information in its local database. In addition, the TOPI interface provides immediate feedback for catastrophic failures to aid debugging. For instance, in Figure 3, TOPI shows the enterprise manager that Ariadne is disabled, along with its assigned role and plan.

4. TEAMCORE: Executing Team Programs

While the enterprise manager uses Karma to specify the team program and monitor its execution, the distributed set of TEAMCORE wrappers, developed in the Soar rule-based integrated agent architecture, perform the actual execution. The STEAM teamwork model [6] provides agents with three forms of domain-independent knowledge of teamwork to enable them to autonomously reason about coordination and communication. *Coherence preserving* rules require team members to communicate with each other to ensure coherent initiation and termination of team plans. *Monitor and repair* rules detect if a team task is unachievable due to unexpected member failure. It then leads the team into reorganization to overcome this failure. *Selectivity-in-communication* rules avoid excessive communication through decision-theoretic communication selectivity.

In the original STEAM implementation, the teamwork knowledge resided directly in the domain agent's knowledge base, an impractical implementation in an open, heterogeneous environment where a domain agent may be a complex legacy system. By placing this knowledge in an external TEAMCORE wrapper, we now no longer need to modify the domain agent itself. However, while the STEAM rules enable the TEAMCORE wrappers to communicate with each other automatically, we now also need a domain-agent interface module to enable a TEAMCORE wrapper to communicate requests to the domain agent it wraps. The interface module allows each TEAMCORE wrapper to send the control and information request messages that are appropriate given the current team activity. The wrapper may then communicate any response from the domain agent to the other TEAMCORE wrappers as part of the usual STEAM procedures.

We have applied our Karma-TEAMCORE framework to the mission rehearsal of the evacuation of civilians from a threatened location. The system designer created a team-oriented program for this problem, using the following agents:

Quickset: (P. Cohen et al., Oregon Graduate Institute) Multimodal command input agents [C++, Windows NT]

Route planner: (Sycara et al., Carnegie-Mellon University) Path planner for aircraft [C++, Windows NT]

Ariadne: (Minton et al., USC Information Sciences Institute) Database engine for dynamic threats [Lisp, Unix]

Helicopter pilots: (Tambe, USC Information Sciences Institute) Pilot agents for simulated helicopters [Soar, Unix]

Although none of these agents had any teamwork capabilities, we successfully used these agents within the Karma-TEAMCORE framework to build a team-oriented program for an evacuation mission rehearsal system. Karma can locate these agents based on the team-oriented program and the specified organization hierarchy. The TEAMCORE wrappers then successfully executed the team-oriented program, consisting of 18 reactive team plans. Even in the face of failures of individual agents, the entire system is robust and does not halt; instead, the team members try to substitute another agent with relevant expertise if possible and/or show graceful degradation. A second aspect of evaluation is measuring the benefit of the TEAMCORE wrappers' domain-independent teamwork knowledge, versus alternative coordination schemes. An alternative would reproduce all of TEAMCORE's capabilities via domain-specific coordination plans, where about 10 separate domain-specific coordination plans would be needed for each team plan. In contrast, with TEAMCORE, we wrote no coordination plans for inter-TEAMCORE communication. Instead, such communications occurred automatically from the team plan specification. A third aspect of evaluation is the ease of modification to the team. For instance, the route planner was the last addition to the team. Its integration required coding of one additional role in the organization, one additional team plan in the plan hierarchy, and the specification of the new agent's capabilities. None of the existing teams, roles, or plans required any modifications.

5. Summary

Collaborative enterprises face daunting challenges when attempting the rapid integration of heterogeneous, distributed components. To this end, this article focuses on enabling designers to rapidly create agent organizations. It describes an agent resources manager, Karma, for assistance in effectively creating and managing agent organizations. As the number and variety of agents available to a particular enterprise increases, new agents, like Karma, that aid in building and maintaining agent organizations will become increasingly critical. This article also focuses on the novel TEAMCORE framework, where teamwork capabilities are built into its very foundations, through the teamwork models in our TEAMCORE wrappers. These wrappers make existing individual domain agents, who are originally not ready to be responsible team members, "team ready". Once made team-ready, these agents enable abstract specifications of an agent organization in the form of team-oriented programs. This can significantly reduce the design effort, since team-oriented programs eliminate the need to script all of the agent interactions. Our framework has shown promise, given its successful application in the concrete collaborative enterprise of the evacuation scenario.

References

[1] Mihai Barbuceanu and Mark S. Fox. The Information Agent: An infrastructure agent supporting collaborative enterprise architectures. In *Third Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises*, 1994.

[2] K. Decker, S. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-97)*, July 1997.

[3] J. Hendler and R. Metzeger. Putting it all together – the control of agent-based systems program. *IEEE Intelligent Systems and their applications*, 14, March 1999.

[4] Michael N. Huhns and Munindar P. Singh. Multiagent systems for workflow. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 8:105–117, 1999.

[5] David L. Martin, Adam J. Cheyer, and Douglas B. Moran. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1-2):92–128, 1999.

[6] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research (JAIR)*, 7:83–124, 1997.