Organizing multi-agent systems for crisis management

Iván García-Magariño, Celia Gutiérrez, and Rubén Fuentes-Fernández

Dept. Software Engineering and Artificial Intelligence
Facultad de Informática
Universidad Complutense de Madrid, Spain
ivan_gmg@fdi.ucm.es, cegutier@fdi.ucm.es, ruben@fdi.ucm.es

Abstract. Crisis management has been a traditional testbed for multi-agent systems concepts on simulation, learning, planning, or communication. Nevertheless, little research has been done in order to gain knowledge from the experience obtained in such scenarios. This paper aims at this goal by describing an organization pattern for multi-agent systems in crisis management, abstracted from several existing case studies in which the agents follow a sequence of interactions and the organization must optimize the use of human resources. The pattern considers an emergent organization of peers that adopt different roles according to the circumstances. The key features of the organization are its robustness, scalability (in terms of both agents and roles), flexibility to deal with a changing environment, and the efficient use of resources. In order to validate the organization, the paper presents its implementation with the INGENIAS methodology. This development follows a model-driven approach, which allows a smooth transition from the specification to the code, and a low-cost testing of the system with different settings.

Key words: crisis management, multi-agent systems, software engineering, design patterns, model-driven development.

1 Introduction

Situations of crisis involving groups of people [5] have been acknowledged as a kind of setting where a proper combination of knowledge-based actions, planning, flexible adaptation, critical analysis, and group interaction are the keys for a successful solution. Given these features, the Artificial Intelligence community, and very specifically the Agent community [7], has considered scenarios involving these situations of crisis as an ideal testbed for their techniques. This kind of testbeds (or scenarios) is generally referred as *crisis management*. Several problems are considered under the term crisis management: search and rescue robots [4]; crowd simulators [8]; contingency management systems [11]; or crisis response [7]. Between all of them, Multi-Agent Systems (MAS) has probably paid more attention to the study of crisis response scenarios.

A crisis response problem considers a catastrophe in which several groups of agents with specific skills must be coordinated to assist to the victims. These

assistant agents collect information of the environment while working and autonomously solve emerging issues. This kind of problem has several features that make of MAS a proper tool for its analysis. First, it has a clear identification of agents, roles and groups. Besides, agents have to interact with a customizable and changing environment. The solution of the problem needs collaboration between the involved agents. Moreover, the scenario offers a clear statement of the goals to satisfy both at the individual and group levels. This relevance has lead to the development of common implementations of the scenario, like RoboCupRescue [7] or DrillSim [3].

As stated before, the crisis response scenarios have been widely used mainly as testbed for concepts of MAS. However, it is also possible to consider the general lessons extracted from the implementations made for this scenario, that is, to try to extract design patterns [6] for MAS from this experience. These patterns can provide useful insights for the design of systems with similar requirements and expected benefits.

This work focuses on the description and implementation of one of such patterns, specifically focused on organizational issues of the MAS. Its main requirements are the following. It considers several groups of agents with different capabilities and goals, mainly citizens and specialists, like medical doctors or firefighters. Each of these groups shares some common goals, although individual persons have specific goals and behave autonomously in the field responding to their specific situations. The use of resources (like communications, movements, or medical supplies) must be planned and optimized as much as possible given their scarcity. However, this version of the pattern does not enforce limitations about distances that agents can travel, available items of supplies, or fails in communications that require the use of ad-hoc networks. Finally, the overall success of the system is regarded as the minimization of the casualties and, if possible, of the damaged infrastructures and properties.

To work in this scenario, a support MAS is organized as follows. People, both citizens and specialists, are supported by agents playing two roles: *InformationR* for those reporting some information or asking for help; and *NetworkR* for optimization of the network use. The reporter role just initiates the communication or processes the results. The networker considers issues such as traffic balance, network fault tolerance, or message addressing to the potentially proper receivers. Both citizens and specialists have a limited radio of communication and knowledge about the situation. To coordinate the different services there are coordinators represented by the role *CoordinatorR*. It gathers the distributed information and makes it available to the central services.

The use of this pattern is shown with the following specific crisis response scenario. A poisonous material has been accidentally released into a city. The number of affected people is very high to be managed only with a centralized solution at hospitals. Besides, the official medical services are overwhelmed to be able to assist affected people at their own locations. Thus, a distributed solution where people on the ground collaborate is necessary. The citizens with medical capabilities (i.e. the specialists) will help affected citizens (i.e. the citizens in the general pattern) who are close enough to them. Citizens will be warned as soon as possible of the infected locations to avoid them. The central official system (i.e. the coordinators) must be informed of all the infected locations. The communications should be efficient, as the communication network is almost collapsed. It is remarkable the fact that, in this scenario, each citizen has been assigned an agent for coordination with other citizens.

Both the description of the pattern and its implementation adopt the state-of-the-art agent-oriented methodology INGENIAS [9] and its support tool the INGENIAS Development Kit (IDK) [2]. INGENIAS covers most of the concepts required for the description of the pattern, with an important focus on the mental state of agents, their interactions, and their organizations. As a drawback, it understates the definition of the space and time features of the environment when compared with the possible requirements of this scenario. INGENIAS uses a model-driven [10] approach for development and this feature facilitates the study of patterns for crisis management. The IDK provides support for modeling, refinement, verification, and code generation with different target platforms. The current case study uses the INGENIAS Agent Framework (IAF) [2] distributed as part of the IDK, and build on top of the JADE platform (available at http://jade.tilab.com/).

The remaining of the paper further explains the pattern and its implementation. Its organization follows. Section 2 describes the requirements of the proposed scenario with more details. After that, Section 3 shows the complete design of the pattern with INGENIAS. The development of the platform specific models that contain the additional information for code generation appear in Section 4. The running system is validated with several tests described in Section 5. Finally, Section 6 discusses the main features of the pattern regarding experimentation and considers some future work.

2 Requirements

The requirements of the pattern are formalized with two use cases that can be seen in Figure 1. These cases are already customized for our specific case study of the poison crisis.

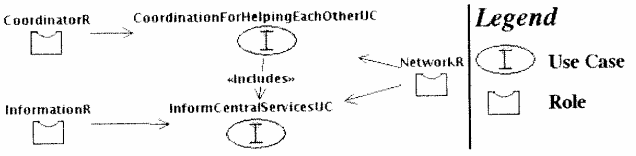

Fig. 1. MainUseCases Diagram.

The first use case is called *CoordinationForHelpingEachOtherUC*. Its goal is the coordination among the people on the ground, that is, citizens and specialists. In this case, specialists are citizens with medical skills, maybe just first aid abilities, who are able to offer help to other citizens. Common people affected by the poison ask for assistance and identify their positions. The second use case is *InformCentralServicesUC*. Its goal is to show how people inform to the central services of the locations affected by the poison. The *InformationR* role participates in this use case. The function of this role is to be the interface of the user with the system and to initiate the transmission of the user's state to the central services.

Note that the *CoordinationForHelpingEachOtherUC* includes the *InformCentralServicesUC* use case. Reporting the infected locations to the central services is part of the coordination activities among people on the ground. This makes possible that the central services can plan assistance for areas with many affected locations. This is a first step for an optimization of the resources available for the management of the poisoning.

In a crisis situation, where the communication network is a critical and probably overloaded resource, it is important to make communications efficient. For this reason, both scenarios include a participant *NetworkR* role for the management and use of the network infrastructure. This role supervises the network and acts as an intermediary in the communications. In order to avoid bottlenecks, the role is played by several agents.

A system for a crisis scenario must also satisfy some key quality requirements. For instance, it must exhibit failure tolerance, good performance, load balance, usability, scalability and testability. This organization considers failure tolerance with redundancy. If a coordinator agent halts, only one citizen cannot access the system but all the others are able to continue communicating. If a network-agent (i.e. an agent playing the role *NetworkR*) fails, the corresponding coordination-agent selects randomly another network-agent to establish the communication. Thus, the whole MAS keeps working without noticing the failure. Note that this organization could also be the basis to support ad-hoc networks. In that case, agents playing the *NetworkR* role would have to locate a device with communication capabilities near their own devices. Finally, if the information-agent (i.e. an agent playing the role *InformationR*) fails, the people on the ground can go on communicating each other, although not with the central services. The other quality features will be discussed in the following sections.

3 Solution

The current MAS pattern is intended to satisfy the requirements mentioned at Section 2. Therefore, its implementation includes three roles that in this case are also assigned to three different types of agents: *coordination-agents*, *network-agents* and *information-agents*. Briefly, the *coordination-agents* are responsible for the coordination among the people on the ground. Each mobile device of the people is habited by a coordination-agent that constitutes the interface with the system. The *network-agents* are in charge of the proper communication among other kinds of agents. The *information-agent* organizes the information about the dangerous locations and shows them to the central official services. The design decisions are further explained within the following sub-sections.

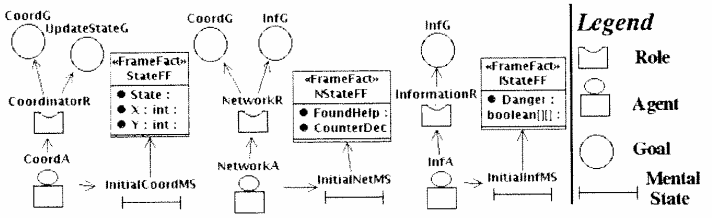

Fig. 2. Role and Agent Definitions of the Presented MAS.

3.1 Roles, goals, and mental states

The responsibilities in the MAS are summarized in Figure 2 with the following roles that agents can play:

- *CoordinatorR*. It is the role of the coordinator and several agents can play it. The goal of the coordination-agents (i.e. *CoordA*) is organizing the people on the ground to help each other and to warn of the poisoned locations. Each user can interact with one coordination-agent.
- *NetworkR*. It is the role that manages the network making its use efficient. Again, several agents can play it. The network-agents (i.e. *NetworkA*) are the intermediaries for the communications. The inclusion of this role allows isolating the potential communication problems in a well-defined container, that is, this role and its capabilities.
- *InformationR*. Several agents can play this role, although the current system only considers one to simplify the discussion. The goal of the information-agent (i.e. *InfA*) is to gather reports of the infected locations and display them. The information-agent uses a city map for visualization.

The tasks assigned to these roles require some information appearing in Figure 2 as mental states. The mental state of a coordination-agent contains information about its user. This state describes the current user's location and state, which can contain the values *need-help*, *can-help*, *being-helped*, *helping*. This information is needed for the cooperation among the agents. For an information-agent, the mental state contains a map with the relevant locations, which in this example are the poisoned ones. The network-agent mental state only contains the facts necessary for the communications.

3.2 Communications

The communications considered in the presented crisis-pattern cover asking-receiving help, warning citizens of dangerous areas and reporting dangerous areas to central services. These communications are collapsed in two interactions, coordination among citizens and coordination with the central services, that are supported by the MAS elements abovementioned.

In all the communications, a network-agent is the intermediary, as explained for the *NetworkR* role. In case of some incidence or change in the characteristics of the network, only network-agents should be modified.

The communication for coordinating citizens happens between coordination-agents and has two goals: asking for help in a dangerous situation and warning citizens of a dangerous location. The same messages are used for both goals. When a citizen needs help, its corresponding coordination-agent sends messages to other coordination-agents through a network-agent. The citizens of the receiver coordination-agents can be either non-qualified citizens or potential specialists. If a citizen can help the requester, the receiver coordination-agent asks its user to go to assist the requester citizen. Otherwise, the citizen cannot help the affected person and the receiver coordination-agent warns its citizen of the dangerous location. Another design option would be to use different messages for each goal. However, this option would increase the number of exchanged messages, hindering the network efficiency.

To reduce further the exchange of messages and the workload of agents, the two communications (among people on the ground and with the central services) share the same initial interaction. Both of them begin with the interaction called Coordinator-Networker. This interaction is initiated from a coordination-agent and sends an asking-for-help (and warning) message to the network-agent. Then, the information-agent initiates two interactions. One interaction completes the coordination between people on the ground. The other interaction carries out the notification of the dangerous location. This simplification also improves the efficiency of the network, as it reduces the number of interactions and consequently the number of messages.

In brief, the improvement of the network efficiency is focused on reducing the number of exchanged messages. For instance, the message raised from a dangerous location accomplishes the three following goals: to ask help, to warn citizens of the dangerous location, and to notify central-services of that location.

Finally, this crisis-pattern also considers the optimization of human-resources to cope with the crisis. When asking for help, it is possible that several persons can assist to the same affected citizen. In this case, the network-agent only asks one of the coordination-agents to come, since it is assumed that only one specialist is needed. For this reason, in the Networker-Coordinator interaction protocol (see Figure 3), the networker (i.e. *NetworkR*) begins asking for help to all the coordinators. Then, all the available coordinators answer whether they can help or not. Finally, the networker only asks one coordinator to come.

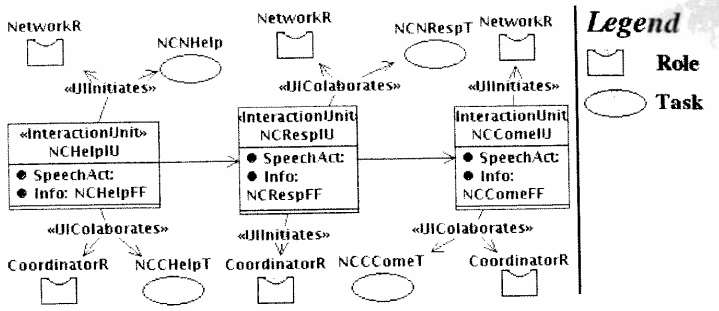

Fig. 3. Networker-Coordinator Protocol. A network-agent searches for help. The networker sends a message to the coordinator-agents. The coordination-agents answer indicating whether they can help or not. If possible, the networker selects one coordinator. The networker asks this coordinator to come.

3.3 Participants in the Interactions

In the interactions of the presented crisis management pattern, the mechanisms to select the participants in each interaction implies crucial improvements of the MAS performance. As it happens with the interactions in the previous sections, the goal is to reduce as much as possible the network traffic.

To begin with, when a coordination-agent asks for help, this agent establishes an interaction with one network-agent. Thus, this interaction is one-to-one and only one network-agent is required. A simple default behavior could be selecting the first agent playing the network role. However, this simple behavior is not the most efficient in a crisis. The first network-agent can collapse while the other network-agents, if any, are idle. To avoid this situation, the policy selected for the presented MAS pattern instantiates the same number of coordination-agents and network-agents. Each coordination-agent is associated with a different network-agent. In case that the coordination-agent cannot contact its associated network-agent, it interacts with another network-agent selected randomly. This policy increases the reliability and the performance of the MAS. The system does not depend on one network-agent, it can work if an agent collapses, and it avoids a potential bottleneck.

Another point of improvement appears when a network-agent searches for a coordination-agent that can help another coordination-agent requesting help. In this situation, the network-agent initiates a one-to-many interaction. A simple default behavior could be involving all the coordination-agents in the interaction. However, this behavior would also involve ask the requester coordination-agent for help, while it should not receive its own message. For this reason, the policy for the Networker-Coordinator interaction is that the network-agent initiates a conversation with all the coordination-agents but the requester.

3.4 Tasks

The tasks of this crisis-management pattern can be classified into several groups. This classification regards the goals related with the execution of these tasks.

A first group of tasks is the responsible of reacting to the events generated by the people on the ground. A person generates events in the interface provided by the corresponding coordination-agent. These events are *need-help*, *can-help*, *update-location* or *none*. The tasks in this group consume these events and update the user's state in his/her coordination-agent mental state. For instance, the task that corresponds to the *need-help* event launches a Coordinator-Networker conversation to ask for help.

Another group of tasks (see Figure 4) is associated to the Coordinator-Networker interaction. These tasks transfer the information about the requester (the agent ID and its location). The most relevant task in this group is the *CNNHelpT* task. This task receives a call for help. It is executed by the network-agent. Its execution launches two different conversations. The goal of the first conversation is to search for help from other coordination-agent; the goal of the second conversation is to inform the information-agent of the new affected location. In addition, within the Coordinator-Networker interaction, some tasks are responsible for bringing back the response to the requester agent. The response indicates whether help has been found or not. If help is found, the response contains the name of the coordination-agent whose user is coming to help the user in danger.

Another group of tasks is associated to the Networker-Coordinator interaction. This interaction is related with the search for help. It begins with the network-agent delivering a frame fact called *NCHelpFF* that contains the ID of the requester agent and its location. This frame fact is sent to request help to all the coordination-agents. The *NCCHelpT* task is associated with the reception of the mentioned frame fact. This task checks out if the user of the receiver coordination-agent agent can help in the related dangerous situation. To do that, the task consults the coordination-agent mental state. Then, if the user has the necessary capabilities, the distance between the requester user and the helper user is calculated. If both users are close enough, the answer is positive. Otherwise, the answer is negative. Another task executed by the network-agent manages all the responses. At most, this task requests one user to come to provide help.

Finally, some tasks are in charge of transferring the new affected location to the information-agent. One of these tasks updates the city map of the affected locations in the information-agent mental state. This city map is shown in the graphical user interface.

4 Implementation

Following the pattern for crisis management presented in previous sections, a complete MAS has been implemented for a poison crisis in which a poisonous

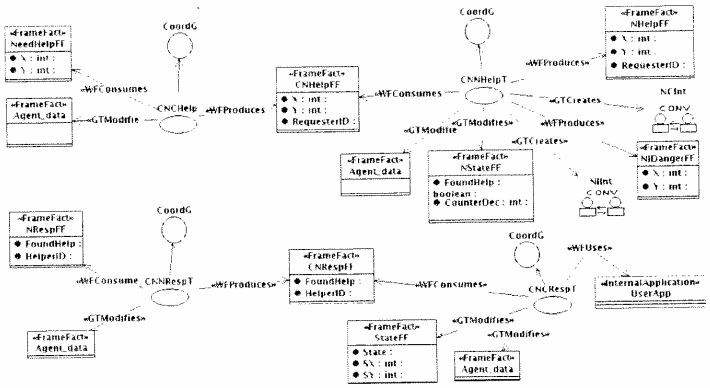

Fig. 4. CNTasks Diagram. Tasks Related to the Coordinator-networker Interaction. When the network-agent receives the help message, two conversations are launched. The first conversation searches for help among other coordination-agents. The second conversation informs the central services. The response to the current coordination agent includes at first instance whether help is found or not. If so, the helper identifier is also included.

material is released in a city. The model-specification and the corresponding software of this MAS is available for MAS practitioners at the *GRASIA* website [1] in the "Training / Full developments examples" section. The implementation skeleton is generated with the INGENIAS Agent Framework (IAF) [2] code generator. Then, the skeleton is manually completed with programming code. The MAS runs on the JADE platform (available at http://jade.tilab.com/), one of the most widespread platforms in the agent community.

Figure 5 shows an example of the MAS execution. In this example, four people on the ground use the MAS. The CoordA_1 and CoordA_3 users have medical capabilities. Thus, both of them press the *Can Help* button. The CoordA_0 user gets affected by the poisonous material so he/she presses the *Need Help* button. The MAS starts to coordinate and inform the central system. CoordA_1 can help CoordA_0 because he/she has medical capabilities and is close enough to CoordA_0 (look at the locations in Figure 5). No other coordination-agent satisfies the same conditions, since CoordA_3 is too far to help CoordA_1. The response is sent back to CoordA_0 with the following message, "CoordA_1 is coming to help you". The other people on the ground are warned of the new poison-affected location with the following message. "Alert! Infection in (0,3). Avoid this location". Finally, the city map of the central services is updated. The new affected location (0,3) is indicated in the visual representation of the map with a different background color.

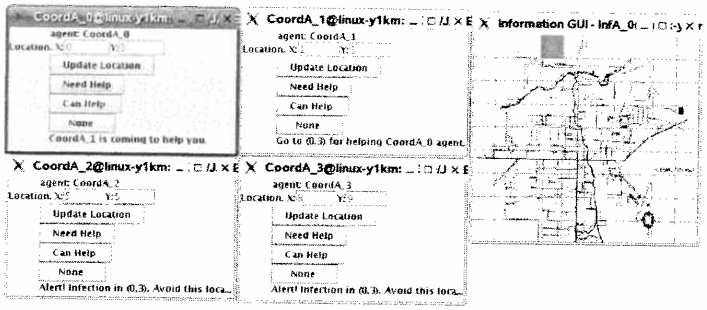

Fig. 5. Example of execution.

5 Testing

The crisis-management pattern has been analyzed through a battery of tests. These tests run over the following deployment configurations.

The first test is called *GeneralTest* and it considers several functionalities of the MAS. First, it checks out that the citizen who is asked to assist an emergency is close enough of the requester and have the required medical capabilities. Second, it also verifies that the requester's dangerous location is notified to the central services and updated in the map at the end of the execution. The initialization of the test is similar to the execution example (see Figure 5) from the implementation of the *release-poison* case study. There are four citizens. The first one requests for help. The second citizen is too far for helping the requester. The third citizen cannot help because of its lack of capabilities. Finally, the fourth one is close enough and has the required capabilities to help. This last citizen is chosen to assist the first one and sent to provide help.

The second test is called *OnlyOneComesTest* and corresponds to the example of deployment *SeveralDoctors* (see Figure 6) taken from the poison-crisis case study. This test is initialized with several people on the ground with help-capabilities. A citizen asks for help and there are several citizens close enough who are potential specialists. Thus, there are several citizens that satisfy the necessary conditions for helping the requester. However, only one potential specialist must be asked to go to assist the requester. As explained in Section 3.2, only one potential specialist is necessary and the other specialists must remain available for future requests. This test also checks the mental state of the coordination-agents at the end of the communication.

6 Conclusions and Future Work

This paper describes a MAS pattern intended for the management of crisis situations where citizens and specialists have to collaborate on the ground. All of

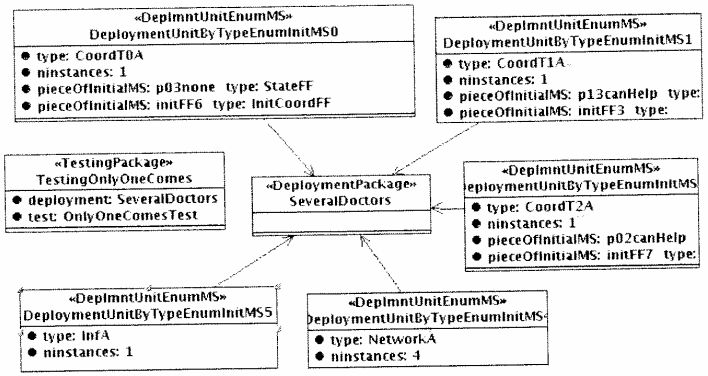

Fig. 6. *SeveralDoctors* deployment for the OnlyOneComesTest test. Three coordinator agents are initialized. Most of them have medical capabilities. They are very close to each other.

them can provide useful information about the state of the crisis but only specialists can solve the emergencies. As key issues of the pattern, it focuses on a right choice of the roles and agents, and an efficient coordination among them. These features are central for this kind of systems that requires a fast response.

Through the paper, a full MAS development has been carried out as basis for the validation of the pattern. The experimental latency measures in communications (which have not been shown here for brevity) show reduced answer intervals with a proper load balance. This efficiency has been gained through a careful design of the MAS. One of the most expensive operations in the presented MAS are the interactions. The agents are located in different mobile devices and their communications depend on the status of the network and other agents. Reducing the number of interactions or participants in an interaction implies an increment of the efficiency in this case. The interactions have also been distributed uniformly among the network-agents to avoid bottlenecks and increase fault tolerance. In addition, the optimization of human-resources is taken into account regarding the number of specialists required to help an emergency, and issues of usability are considered through an adequate user interface.

For future work, several goals rise. Firstly, crisis management cases are subjected to constraints of resources and location, i.e. when an agent needs help it is necessary to locate the nearest agents that can help it. Consequently an intelligent scheduling will make the system more efficient, although it implies a need of exchanging more information between the agents. Scheduling can also be applied to the requests of help in each agent. These features will be implemented in the tasks. Secondly, the designer must implement a solution that prevents undesirable behaviors of the MAS. One of these behaviors appears when there

are agents that keep on sending messages to require a service when they do not have an immediate response. Other happens when agents are so saturated with received messages that their response latencies become unacceptable high. This communication patterns cause unbalanced MAS with a low quality of service in terms of response times. Metrics to detect these patterns will be incorporated to the tasks that participate in conversations. In this way, it will be possible to discover the patterns that correspond to efficient and inefficient systems. Finally, the system will contemplate the damages in the communication network. In such cases, ad-hoc networks can be a way to offer communication without external infrastructure, just using the capabilities of the citizens' portable devices. For this scenario, the *NetworkR* role will incorporate mechanisms to detect such situations and to use alternative networks instead.

Acknowledgements. This work has been supported by *Methods and tools for agent-based modeling* project, supported by Spanish Council for Science and Technology with grant TIN2005-08501-C03-01; and Grant for Research Group 910494 by the Region of Madrid (Comunidad de Madrid) and the Universidad Complutense Madrid.

References

1. GRASIA web. `http://grasia.fdi.ucm.es/`.
2. INGENIAS Development Kit. `http://ingenias.sourceforge.net/`.
3. V. Balasubramanian, D. Massaguer, S. Mehrotra, and N. Venkatasubramanian. DrillSim: A Simulation Framework for Emergency Response Drills. *Proceedings of the IEEE International Conference on Intelligence and Security Informatics. Lecture Notes in Computer Science*, 3975:237–248.
4. A. Davids. Urban search and rescue robots: from tragedy to technology. *IEEE Intelligent Systems*, 17(2):81–83, 2002.
5. Y. Engestrom. Perspectives on Activity Theory, 1999.
6. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. 1995.
7. H. Kitano. RoboCup Rescue: a grand challenge for multi-agent systems. *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pages 5–12, 2000.
8. Y. Murakami, K. Minami, T. Kawasoe, and T. Ishida. Multi-agent simulation for crisis management. *Knowledge Media Networking, 2002. Proceedings. IEEE Workshop on*, pages 135–139, 2002.
9. J. Pavón, J. Gómez-Sanz, and R. Fuentes. The INGENIAS Methodology and Tools. *Agent-Oriented Methodologies. Idea Group Publishing*, pages 236–276, 2005.
10. D.C. Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *COMPUTER*, pages 25–31, 2006.
11. L.B. Sheremetov, M. Contreras, and C. Valencia. Intelligent multi-agent support for the contingency management system. *Expert Systems With Applications*, 26(1):57–71, 2004.

Analysis and Design of a Multi-Agent System Using Gaia Methodology in an Airport Case of Use [1]

Nayat Sánchez-Pi, Javier Carbó, José Manuel Molina

Computer Science Department. Carlos III University of Madrid
Av. de la Universidad Carlos III, 22 Madrid, Spain

{Nayat.Sanchez, Javier.Carbo, Jose.Molina}@uc3m.es

Abstract. Progress in software engineering has been made through the development of natural high-level abstractions with which to model and develop complex systems. The abstraction and modeling of agent technology play an important role in distributed systems. Multi-Agent System paradigm introduces a number of new abstractions when compared to more traditional systems. They may be used by software developers to more naturally understand, model and develop an important class of complex distributed systems. Accordingly new analysis and design methodologies are needed to effectively engineer such systems. In this paper, we propose the analysis and design of a multi-agent system for the provisioning of context-aware services in an ambient intelligence domain: an airport.

Keywords: Multi-agent systems, methodologies, context-aware

1 Introduction

Ambient Intelligence is the paradigm to equip environments with advanced technology to create an ergonomic space for the user where they could interact with their digital environments the same way they interact with each other. It is also associated to a society based on unobtrusive, often invisible interactions amongst people and computer-based services taking place in a global computing environment. Services in AmI will be ubiquitous in that there will be no specific bearer or provider but, instead, they will be associated with a variety of objects and devices in the environment, which will not bear any resemblance to computers. People will interact with these services through intelligent and intuitive interfaces embedded in these objects and devices, which in turn will be sensitive to what people need. Another good point seen on the AmI vision is that the electronic or digital part of the ambience (devices) will often need to act intelligently on behalf of people. The components of ambience will need to be both reactive and proactive, behaving as if they were agents that act on behalf of people. If we assume that agents are abstractions for the

[1] Funded by projects CICYT TSI2005-07344, AUTOPIA (IMSERSO) and CAM MADRINET S-0505/TIC/0255