Engineering CoMMA Multiagent System with Agent UML

Agostino Poggi, Giovanni Rimassa, and Paola Turci

Abstract—**This paper is an attempt to engineer a practical multiagent system by using Agent UML. We focus our attention on CoMMA system, an open, agent-based system for the management of a corporate memory. The main objectives of the paper are: to prove that Agent UML can be successfully applied to real-world applications, to discover what is missing and finally to propose improvement in order to capture aspects that AUML has not considered so far.**

Index Terms—**AUML, Multiagent System, Software Engineering, UML.**

I. Introduction

Software engineering methodologies were and are essential, especially in industry, to improve software productivity, lowering the costs and ensuring a high level of quality.

The research on agent-oriented software engineering is based on the possibility to model a software system at the agent level of abstraction [13][12][11][4][5][7]. This level of abstraction considers agents as atomic entities that communicate to implement the functionality of the system.

As evinced from an analysis of the majority of the proposed languages and methodologies, despite the efforts spent in defining new metaphors, symbols and diagrammatic notations, the modelling languages proposed are in general quite powerful but they still remain incomplete. In this paper we consider Agent UML [11], a modelling language based on UML. As UML, AUML is not a methodology by itself, but a suggested notation to be used within the framework of a methodology. AUML mainly concerns agent class diagram and protocol diagram so far seen. Other UML diagrams could be revisited in the future in order to cope with the special requirements of multiagent systems, or some new diagrams might be included in AUML to tackle particular concepts not present in UML.

Even if a strong interest is manifest on Agent UML, very few applications of Agent UML to a real-world exist at present [9]. The aim of this paper is to use Agent UML to engineer CoMMA multiagent system in order to verify the expressive power of this modelling language and eventually propose improvement in order to capture aspects that Agent UML has not considered so far.

CoMMA [6] is an innovative and quite powerful software system for knowledge management in corporate networks. CoMMA is ontology centred and exploits intelligent agents to produce, deliver and search for semantic-level information in a flexible and scalable way.

The approach used in the CoMMA project in the design of the multiagent architecture was a very simplified agent-oriented engineering approach. This is because agent-oriented engineering was a new research topic and most of the actual agent oriented modelling languages and methodologies were work in progress during the development of CoMMA project.

This paper describes how the overall architecture of the CoMMA multiagent system can be engineered by using Agent UML; in particular this paper deals with issues such as agent role and task identification, acquaintance and interaction specification, configuration and deployment.

The next section illustrates an overview of the CoMMA system and its main objectives. Section three shows the application of the Agent UML diagrams to the CoMMA system. In particular the first subsection describes the general architecture of CoMMA and introduces the different kinds of agent that populate the CoMMA societies. The second subsection shows how single agents can be described with AUML agent class diagrams. The third subsection, using AUML protocol diagrams, shows how the interaction of each agent with the other agents can be analysed in order to implement the CoMMA scenarios. The fourth subsection deals with the issues of the configuration and deployment and a proposal of an agent deployment diagram is illustrated. Finally, the fourth section concludes with a discussion about Agent UML features.

This work is supported in part by the European Commission through the contract IST-2000-28385, *Agentcities.RTD*.

A. Poggi is with the AOT Lab - Dipartimento di Ingegneria dell'Informazione - Università degli Studi di Parma (e-mail: Poggi@CE.UniPR.IT).

G. Rimassa is with the AOT Lab - Dipartimento di Ingegneria dell'Informazione - Università degli Studi di Parma (e-mail: Rimassa@CE.UniPR.IT).

P.Turci is with the AOT Lab - Dipartimento di Ingegneria dell'Informazione - Università degli Studi di Parma (phone: +39 0521 905708, fax: +39 0521 905723, e-mail: Turci@CE.UniPR.IT).

II. CoMMA overview

CoMMA (Corporate Memory Management through Agents), a FIPA compliant agent system implemented through the use of JADE agent development software framework, is the result of an international project funded by European Commission.

In the CoMMA project we were interested in implementing and evaluating a corporate memory management framework based on several emerging technologies: agents, ontology engineering and knowledge modelling, XML, information retrieval and machine learning techniques. The project addresses particularly the problem of *information retrieval* faced by employees involved in the following scenarios:

- Enhancing the insertion of new employees in the company,
- Performing processes that detect, identify and interpret technology movements and interactions for matching technology evolutions with market opportunities to disseminate among employees innovative ideas related to technology monitoring activities.

The main objective was that the final system should help users retrieve information relevant for them: details of the technology underlying the agent and the resources the agent accesses are 'abstracted' away - that is, they are user-transparent. The agent enables a person to state what information he or she requires; the agent determines where to find the information and how to retrieve it.

The technical choices made in CoMMA were mainly motivated by three observations:

- The corporate memory is, by nature, an heterogeneous and distributed information landscape
- The population of the users of the memory is, by nature, heterogeneous and distributed in the corporation. Agents will be in charge of interfacing and adapting the system to the user.
- The tasks, as a whole, to be performed on the corporate memory are, by nature, distributed and heterogeneous.

III. Application of agent UML diagrams

A. Agent Decomposition of CoMMA System

CoMMA system uses agents for wrapping information repositories (i.e., the corporate memory), for the retrieval of information, for enhancing scaling, flexibility and extensibility of the corporate memory and to adapt the system interface to the users. These tasks are performed through the cooperation among different kinds of agents that can be divided in four sub-societies:

1. Document and annotation management.
2. Ontology (Enterprise and User Models) management.
3. User management.
4. Agent interconnection and matchmaking.

The agents from the document dedicated sub-society are concerned with the exploitation of the documents and annotations composing the corporate memory. They will search and retrieve the references matching the query of the user with the help of the ontological agents. The agents from the ontology dedicated sub-society are concerned with the management of the ontological aspects of the information retrieval activity especially the queries about the hierarchy of concepts and the different views. The agents from user dedicated sub-society are concerned with the interface, the monitoring, the assistance and the adaptation to the user. Finally the agents from the interconnection dedicated sub-society are in charge of the matchmaking of the other agents based upon their respective needs.

The configuration issues outlined above affect every sub-society of the CoMMA multi-agent system; there are many different social models that can be used. A way to distinguish and compare them is to tell what aspects are homogeneous across every member of the sub-society and what other ones are heterogeneous. This can be done with respect to the skills (i.e. the information processing capabilities) or to the content (the information processed). A homogeneous aspect will be replicated in every agent of the sub-society, whereas a heterogeneous one is present at different levels in different agents. The internal organization of each sub-society is mainly the result of the various configuration issues.

The subsequent organizational structures were chosen for each sub-society:

- ***Ontology dedicated sub-society***. Since the ontology is of limited size, it should not be evolved much and since most of the queries will need the whole ontology to apply inference algorithms, the replication society was used. There will be many *Ontology Archivist* agents, and each one of them will have a complete copy of the ontology.
- ***Document dedicated sub-society***. This sub-society handles documents and their annotations that together compose a large database. The hierarchical society was chosen. The various annotation bases will be handled by *Annotation Archivist* agents, but user queries will be handled by *Annotation Mediator* agents.
- ***Connection dedicated sub-society***. The CoMMA multiagent system relies on standard *Directory Facilitator* agents for finding services.
- ***User dedicated sub-society***. Each user of the CoMMA system will have its profile stored in the user home location and managed by a *User Profile Archivist* agent; it will be deployed according to where users belong. To assists the user and to present the results, there will be an *Interface Controller* agent associated to each user logged into the system. The *User Profile Manager* will handle all the users logged in a particular location; it will be

deployed according to where users are currently logged.

B. Agent Level Design

The first phase of the system design, as illustrated in the previous section, is responsible to identify the different kinds of agents involved in the agent society of the MAS, divide them in sub-societies, and describe how such sub-societies can be organized.

The subsequent step is to describe each agent of the MAS using the *Agent Class Diagram* [3][8]. For each agent are identified: its role, its responsibilities and expertises, the agents interacting with it and the coordination and negotiation protocols used in such interactions.

<<agent>>
Annotation Mediator/AM
Organization
Document sub-society: Annotation Mediator
Capabilities
- Provide an access to the document sub-society for information storage and retrieval purposes, managing a set of *Annotation Archivist* agents. - Provide notifications for each new annotation and re-edited annotation to enable proactive information pushing.
Protocols
- Initiates a *FIPA-Contract-Net* protocol to answer user adding annotation requests. - Initiates a *FIPA-Request* protocol to register itself with a *Directory Facilitator*. - Initiates a *FIPA-Request* protocol to deregister itself with a *Directory Facilitator*. - Initiates a *FIPA-Query* protocol with a set of *Annotation Archivist*. - Responds to *FIPA-Request* protocol with a U*ser Profile Manager* to update the corporate memory. - Responds to *FIPA-Query* protocol with CoMMA ontology content
Collaborators
- Directory Facilitator - Annotation Archivist - User Profile Manager
Expertise
- CoMMA-ontology, expressed in RDF Schema and maintained by *Ontology Archivist*. - FIPA-Agent-Management ontology

Fig. 1: Annotation Mediator agent class diagram

This phase starts from the results of the requirement analysis phase and, in particular, from the use cases defining the application scenarios of the system. In the following subsection we illustrate how the procedure is applied to a particular agent, that is *Annotation Mediator* agent.

1) Annotation Mediator Agent

The Annotation Mediator has the duty to manage query about the CoMMA corporate memory content.

Only this agent has access to the enterprise corporate memory. The Annotation Mediator interacts with the agents collaborating in the searching of annotations, that is, the *Directory Facilitator*, *Annotation Archivist* and *User Profile Manager* agents. The interaction with these agents is done via exchange of ACL messages within FIPA query, contract net and request protocols.

The agent class diagram, giving a detailed view of the *Annotation Mediator* agent, is shown in Fig. 1.

As you can see in Fig. 1, the AUML agent class diagram is slightly modified. In fact, on the basis of our experience we believe that it is useful to add two more elements to agent class diagram, in particular:

- An element, called "collaborators", which contains information regarding the agent acquaintances, that is a list of agents with which the agent itself could collaborate.
- An element, called "expertise", to describe the ontologies, which can be used by the agent during its communications with other agents.

C. Agent Interactions

This phase defines the interactions between agents in order to implement the MAS use cases defined in the requirement analysis phase. The agent interaction is the most important behavioural abstraction and the main part of it is the temporal trace of the messages exchanged among agents during their conversation. For each interaction, a natural language description similar to a textual use case scenario is given, and a protocol diagram, using AUML *Protocol Diagrams* [2], is drawn.

To completely define an interaction among FIPA compliant agents, we decided to provide the content of the major message slots. Therefore, for every ACL message exchanged, the following information is present:

The FIPA ACL performative of the message (request, inform, etc.).

1. The interaction protocol ruling the conversation of this message (if any).
2. The message content.
3. The content language used to express the message content.
4. The ontology defining the concepts used in the content.

In the following subsection it is illustrated how the process is applied to a particular scenario, that is "add new annotation" scenario.

1) Add New Annotation Scenario

This scenario is played whenever the user wants to add a new RDF annotation to an existing document base; the purpose of this scenario is to insert a new RDF annotation into the corporate memory.

The protocol diagram, giving a detailed view of the "add

new annotation" scenario, is shown in Fig. 2.

Below a textual description of the scenario is given.

1. The *Interface Controller (IC)* queries the *Ontology Archivist* (*OA)* for the concepts the user is browsing during the composition process. Many conversations ruled by the *fipa-query* protocol take place.
2. The *IC* requests its *User Profile Manager (UPM)* to put the completed annotation into the corporate memory. A conversation ruled by the *fipa-request* protocol takes place.
3. The *UPM* requests the *Annotation Mediator (AM)* to put the completed annotation into the corporate memory. A conversation ruled by the *fipa-request* protocol takes place.
4. A conversation ruled by the *fipa-contract-net* protocol takes place, where the *AM* finds out which member of the *Annotation Archivist*s team will store the new annotation.

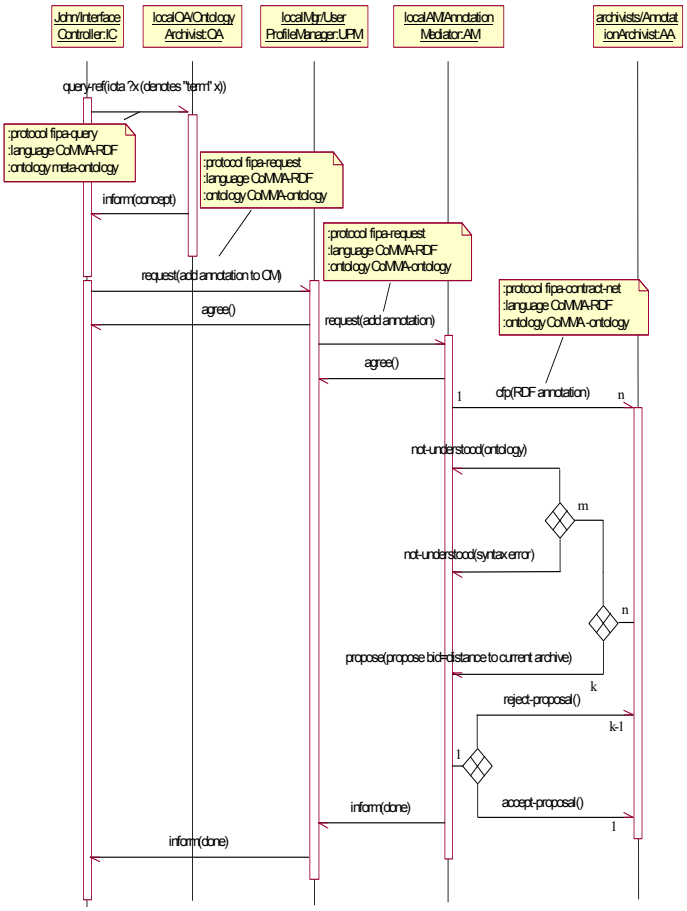

Fig. 2 : "Add new annotation" scenario protocol diagram

At this point the design phase is completed; the next step is the implementation phase. The procedure, which could be applied in the next phase, depends on the framework one intends to use. If the choice falls on JADE framework, the procedure is quite straightforward. In fact the design documents generated using our approach are detailed enough to have a natural implementation with JADE; each use case, defined by a textual description and protocol diagrams, can be easy mapped into a set of JADE behaviours

D. Configuration and Deployment Diagram

The deployment diagram is useful to model highly distributed MAS, that is systems in which it is important to visualize the system's current topology and distribution of components and to reason about the impact of changes on the topology.

The role of the deployment diagram in UML is to describe the position of the components in the execution environment. In the case of a MAS this means to represent hosts (servers, front-ends, etc.), resources, physical agents and their acquaintance graphs, and, depending on the framework used in the implementation, MAS platforms (see Fig. 3).

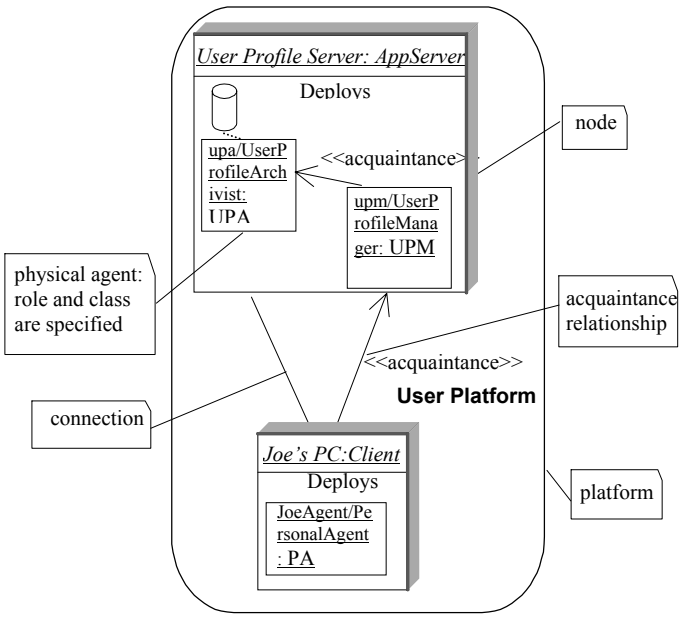

Fig. 3: MAS deployment diagram

An important observation to be made at this point is the difference that exists between the architecture of a MAS and its configuration at deployment time. The architecture of a MAS is a structure that portrays the different kinds of agents of its society and the relationships among them. The architectural description is studied and fixed when designing the MAS. A configuration is an instantiation of an architecture with a chosen arrangement and an appropriate number of agents. One frozen architecture can lead to several configurations. The configuration is tightly linked to the topology and the context of the place where it is rolled out. The architecture is designed so that the possible configurations cover the different system organizational layouts foreseeable in the context of the project. Agent can be arranged among various machine configurations in order to better use available processing power and network bandwidth.

The deployment of a MAS system therefore is driven by:

- The system organizational layout: the structure of

the company, etc.

- The network topology: technical environment and its constraints such as the topologies characteristics, the network maps and data rates, data servers location, gateways, firewalls, etc.
- The interests area: where are the stakeholders (users, system managers, providers, etc.)

We believe that to study and document the configuration description of a MAS at deployment time is necessary to use adapted UML deployment diagrams. In fact the UML deployment diagram is not suitable to illustrate the deployment diagram of a MAS; few modifications are necessary in order to represent new entities like physical agents, their acquaintances and in some cases MAS platforms.

Agent UML has not tackled this kind of diagram yet. As to this subject, in the following we show our proposal of a multiagent system deployment diagram.

A deployment diagram is a graph of nodes connected by communication associations. In the case of MAS nodes may contain physical agents; this indicates that the agents runs or executes on the node. Agents are connected to other agents by dashed-arrow links that represent acquaintance relationship. This indicates that one agent could communicate with the "known" agents by means of interactions protocol.

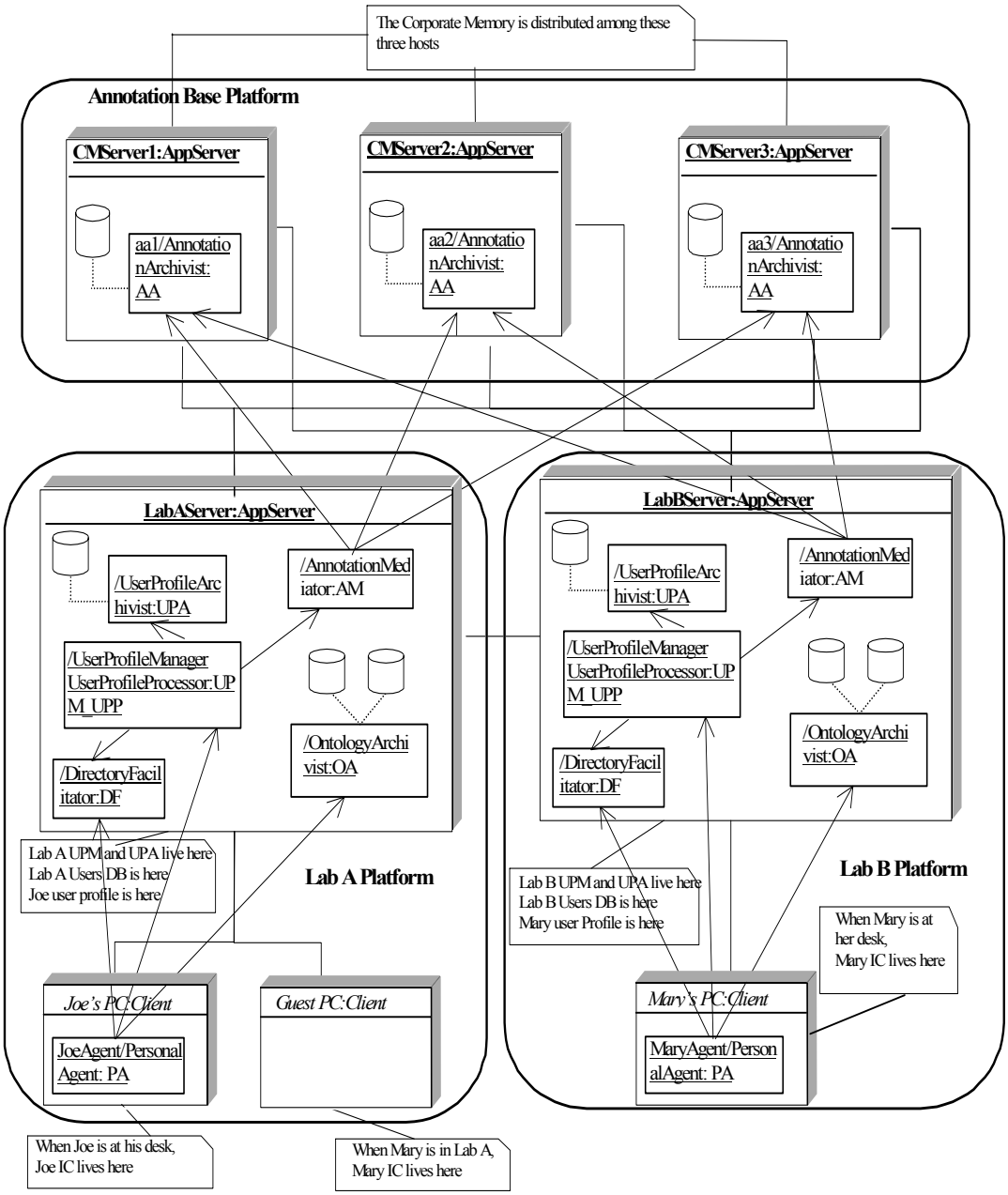

Fig. 4: Three-Tier Deployment with everybody home

The following basic elements are therefore used within multi-agent system deployment diagram:

1. Nodes
2. Connections
3. Physical agents and agent roles
4. MAS platforms
5. Acquaintance relationships

A node, as in UML deployment diagram, is an element that exists at run time and represents physical hosts, that is a computational resource, generally having some memory and processing capability, on which agents may be deployed. Graphically a node is rendered as a cube. Every node instance has a name, a textual string, and a type. The most common kind of relationships among nodes are associations that represent physical connections.

1) Three-Tier Deployment Scenario

The Fig. 4 shows a possible deployment, called *Three-Tier Deployment* because the physical computers hosting the agents are divided into clients (*client tier*), application servers (*middle tier*) and database servers (*DB tier*). Such a deployment strategy has the main advantage of matching one of the most popular intranet structures.

The agent acquaintance graph shows the case of two users logged into the system. The first user, named *Joe*, belongs to the *Lab A* organizational unit and is currently sitting in front of his PC. The second one, named *Mary*, belongs to the *Lab B* organizational unit and is currently sitting in front of her PC.

According to the three-tier architecture, only *Interface Controller* agents live on the client machines (performing pure presentation) and only *Annotation Archivist* agents live on the database servers (performing pure data management). The middle tier, as in classical three-tier architectures, connects the client tier and the data base tier (through *DF* and *AM* agents) and is where the most interesting stuff happens (ontology and user model management). In this deployment, the ontology and the user model belong to the middle tier and not to the data base tier because they are supposed to be much smaller in size than the annotation base and they are not distributed.

In this deployment example, there is a JADE platform spanning the whole *Lab A*, another platform spanning the *LabB* and a third platform that contains all the *Annotation Archivist* agents managing the distributed annotation base. This means that any agent within the *Lab A* has an a priori knowledge of the *Directory Facilitator* living on the *Lab A Server* machine, so registering the *User Profile Manager*, the *Ontology Archivist* and the *Annotation Mediator* with the local *DF* is enough to connect the user sub-society and the ontology sub-society. Moreover, all those connections are local to the *Lab A Server* computer.

The user dedicated sub-society is deployed according to the following rules:

- The *Interface Controller* is active for a session and runs on the host where the user logs for the current session.
- The *User Profile Manager* is active continuously and runs on a server that manages the *current location* of the user (i.e. the machine he or she is logged on).
- The *User Profile Archivist* is active continuously and runs on a server that manages the *home location* of the user (i.e. the machine where he or she belongs and where he or she usually logs on).

IV. Discussion and Conclusion

The aim of Agent UML is to extend and apply a widely accepted modelling and representation formalism, that is UML. So far very few empirical tests have been carried out to evaluate the benefit of Agent UML framework. From our experience we see concrete advantages in the use of AUML. On the one hand Agent UML is a good approach as it gives the opportunity to capitalize on the skills of designers; indeed multiagent system developers are often software engineers who use UML or are aware of it. On the other hand Agent UML models turned out very useful to represent a static view of the system and the interactions between the agents populating the CoMMA societies. The application of AUML agent class diagram and protocol diagram to CoMMA multiagent system, however, shows the need to carry out little improvement. Indeed in these models the aspect of semantic communication has not been considered. We estimated opportune to slightly modify the models by adding information related to ontologies and content languages used by agents. Moreover we realized that the application of UML deployment diagram to MAS is not suitable. We believe that the role of the MAS deployment diagram is to represent hosts (servers, front-ends, etc.), resources, but also physical agents, their acquaintance graphs, and, depending on the framework used in the implementation, MAS platforms. In this paper we made a proposal of an AUML deployment diagram, that is an UML deployment diagram enhanced with agent based concepts, and we use it to describe the deployment of the CoMMA system.

Acknowledgment

Thanks to all CoMMA partners who contributed to the development of the project.

This work is partially supported by the European Commission through the contract IST-2000-28385, *Agentcities.RTD*.

References

[1] Agent UML - AUML Home Page. Available at http://www.auml.org.

[2] Bauer B., J. P. Müller, J. Odell. *Agent UML: A Formalism for Specifying Multiagent Interaction* Agent-Oriented Software Engineering, Paolo Ciancarini and Michael Wooldridge eds., Springer, Berlin, pp. 91-103, 2001

[3] Bauer B., *UML Class Diagrams: Revisited in the Context of Agent-Based Systems*, In Proc. of Agent-Oriented Software Engineering (AOSE) 2001, pp.1-8, Agents 2001, Montreal

[4] Caire, G., P. Chainho, R. Evans, F. Garijo, J. Gomez Sanz, P. Kearney, F. Leal, P. Massonet, J. Pavon, J. Stark *Agent Oriented Analysis using MESSAGE/UML*, In Proc. Second International Workshop on Agent-Oriented Software Engineering (AOSE-2001), Montreal, Canada, May, 2001, 101-107.

[5] Ciancarini P., M.J. Wooldridge, *Agent-Oriented Software Engineering, Lecture Notes in Computer Science*, 1957, Springer-Verlag, 2001.

[6] CoMMA Project Home Page. Available at http://www.ii.atosgroup.com/sophia/comma/HomePage.htm.

[7] Giunchiglia F., J. Mylopoulos, A. Perini. *The Tropos Software Development Methodology: Processes, Models and Diagrams* Proc. AAMAS Conference, Bologna, 2002.

[8] Huget M.-P., *Agent UML Class Diagrams Revisited*. In Proc. of Agent Technology and Software Engineering (AgeS), Bernhard Bauer, Klaus Fischer, Jorg Muller and Bernhard Rumpe (eds.), Erfurt, Germany, October 2002

[9] Huget M.-P., *An Application of Agent UML to Supply Chain Management*. In Proc. of Agent Oriented Information System (AOIS-02), Paolo Giorgini and Yves Lespérance and Gerd Wagner and Eric Yu (eds.), Bologna, Italie, July 2002.

[10] MESSAGE Consortium, *"MESSAGE: Methodology for Engineering System of Software Agents"*, deliverable of the EURESCOM Project P907, 2001.

[11] Odell J., H. van Dyke Parunak, and B. Bauer. *Extending UML for agents*. In G. Wagner, Y. Lesperance, and E. Yu, editors, Proc. of the 2nd Int. Workshop on Agent-Oriented Information Systems, Berlin, 2000. iCue Pub-lishing.

[12] Wooldridge M., N.R. Jennings, D. Kinny. "*The Gaia Methodology for Agent-Oriented Analysis and Design*". Kluwer Academic Press, 2000.

[13] Wooldridge M. and N.R. Jennings, "*Agent-oriented software engineering"*. In J. Bradshaw, editor, Handbook in Agent Technology, MIT Press, 2000.