From Roles Modeled by Using the MESSAGE Methodology to their Implementation with the ASF Framework

Carolina Howard Felicíssimo, Carlos José Pereira de Lucena and Viviane Torres da Silva

Departamento de Informática – PUC–Rio
Rua Marquês de São Vicente 225, Gávea – 22.451-900 – Rio de Janeiro – RJ – Brazil
{cfelicissimo, lucena, viviane}@inf.puc-rio.br

Introduction

In multi-agent systems (MAS) roles are an important issue since they define responsibilities and obligations to the agents, i.e. they guide and restrict the behaviors of agents in a social context [Silva et al., 2004b]. Several agent oriented software methodologies such as MESSAGE [Caire et al., 2002], GAIA [Wooldridge et al., 2000] and TROPOS [Mylopoulos et al., 2001], modeling languages such as MAS-ML [Silva and Lucena, 2004a] and AUML [Odell et al., 2000], and software architecture such as JADE [Bellifemine et al., 2001] and ASF [Silva et al., 2004b] concern with agent roles.

Although roles are being used in developed MAS, there is still a need for case studies that demonstrate how to define roles in the context of organizations and how to assign agents to roles.

This paper presents a case study of the urban traffic domain that illustrates the use of roles during the analysis phase as well as the mapping of the generated role models into code. While using the urban traffic domain, we are interested in provide a semantic support to the governance of laws. In the attempt to enable law conscious among software agents, an ontology was used to describe the semantics of the laws that regulate MAS. The case study was modeled by using the MESSAGE methodology and was implemented with the ASF framework.

MESSAGE Methodology

MESSAGE is a methodology for agent oriented software engineering that extends UML [UML, 2005] with "knowledge level" agent-oriented concepts. In this level, the role concept is presented as a concrete entity.

The methodology covers MAS analysis and design phases and it is easy to learn because it has a well-defined process for its two specific phases. In the analysis phase, Message provides a notation with very simple concept symbols and relations to create its diagrams, enabling focus in different aspects. From the analysis to the design, Message defines a process and UML diagrams are normally used to represent the product of the last phase.

In this paper, we will only consider the representations of the analysis phase. In this phase, MESSAGE uses a number of views or perspectives that emphasize different aspects of MAS such as autonomy, interaction and adaptation. Each view concentrate in a limited but consistent aspect and together they provide a comprehensive view. Diagrams are used to represent specific views, and roles can be modeled in most of them. For instance, in the organization view (acquaintance relationships diagram) roles played in organizations are identified and in the goal/task view (workflow diagram) tasks are associated with roles.

Agent Society Framework

The Agent Society Framework (ASF) is an object-oriented framework designed for implementing agent societies using JAVA. It defines agent applications' architectures that support the implementation of agents, roles, organizations and environments as first-order abstractions. However, those entities are not abstractions available in object-oriented systems. Because of that, it is necessary to define a set of classes that embodies an abstract design for the new entities. The framework is composed of sets of object-oriented modules and their relationships. Each module represents an entity by mapping the entity into a set of classes and relationships [Silva et al., 2004b].

The *Agent Role* module defines the agent role entity by an abstract class called *AgentRole*. This class is associated with others, which correspond to goals (*Goal* class), beliefs (*Belief* class), duties (*Duty* class), rights (*Right* class) and protocols (*Protocol* class) – properties of agent roles. Fig.1 illustrates all the described classes from the module and their associations. The *Action* class not belongs to the module, but it is also illustrated in the Fig.1 because it has relationships with the *Right* and *Duty* classes.

To instantiate the agent role module, the abstract classes *AgentRole* and *Protocol* have to be extended and instantiated. After that, instances of the concrete classes *Right*, *Duty*, *CompositeBelief*, *LeafBelief*, *CompositeGoal*, *LeafGoal* and *Message* have to be created in order to represent the agent role's permissions, agent role's obligations, agent role's beliefs and agent role's goals, and to associate them to the agent role instance created

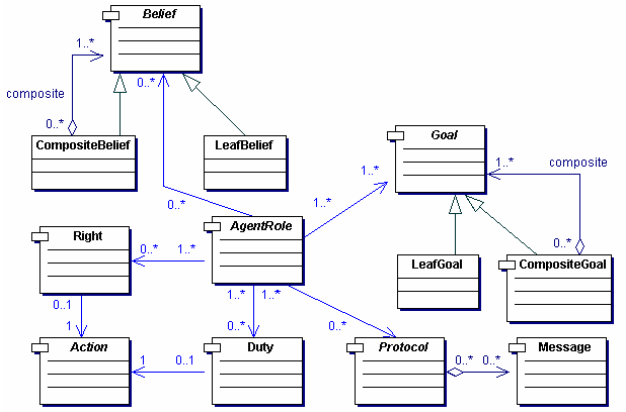

Fig.1. Agent role module, from [Silva et al., 2004b].

Case Study

Urban Traffic Simulator System (UTSS) is an open multi-agent system with a semantic support to the governance of laws. UTSS represents an urban environment that simulates the behavior of three agent's roles: car driver, police officer and pedestrian. Besides, there are other elements as traffic signs, urban paths and places.

An urban traffic's scenario is illustrated in Fig.2. The proposed environment is composed (i) by agents playing the car driver role, the pedestrian role and/or the police officer role; (ii) by four types of traffic signs (traffic signals, speed limit control signs, one-way signs and two-way signs); (iii) by five numbered urban paths and (iv) by the four places: church (accessed via path 1), house in the town (accessed via paths 2 and 4), house on the mountains (accessed via path 3) and hospital (accessed via path 5).

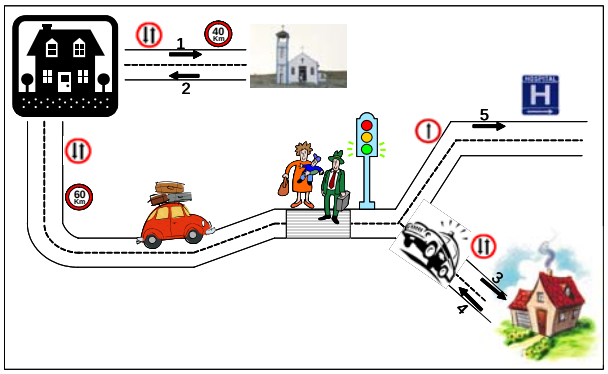

Fig.2. An urban traffic's scenario

Car drivers and pedestrians go and come from different places by using urban paths, which have several traffic signs. According to their goals, car drivers and pedestrians choose to obey or not the traffic sign rules. Police officers monitor the traffic signs according to their laws and can apply penalties for disobedient active entities. A traffic sign can have zero or many police officers monitoring it.

UTSS's Ontology

To provide consciousness of agents on MAS, a semantic support is desired. This type of support can be given by ontologies, making the represented information of a domain easier for machines to automatically process their meaning [McGuinness and Harmelen, 2004]. Ontologies are conceptual models that embody shared conceptualizations of a given domain [Gruber, 1993]. Ontologies languages are designed for use by applications (machines) that need to process the content of information instead of just presenting information to humans [Smith et al., 2004].

The UTSS case study has an ontology describing the inactive entities of the proposed urban traffic environment. All traffic signs, urban paths and places, as well as norms and their associated penalties are described in UTSS's ontology, illustrated in Fig.3.

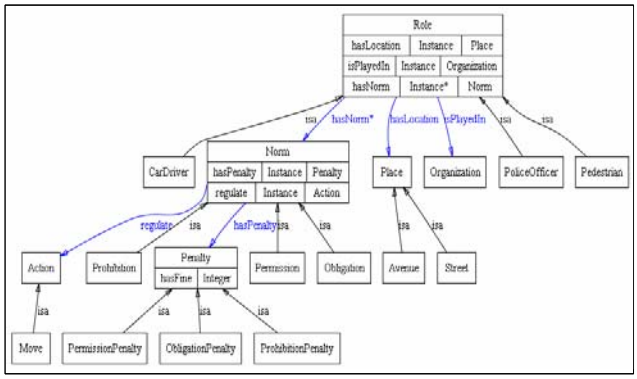

Fig.3. The UTSS's ontology

Active entities from the UTSS access its ontology as a resource in order to get, for example, the traffic signs' meanings and the roles' norms. For instance, when an agent playing the car driver role perceives the green light (an attribute) of a traffic signal (a concept) it accesses the UTSS's ontology to know how to act. Green lights can address move or stop actions. According to the law described by traffic signs' meanings and the agent roles' norms, the agent can choose to drive compliant or not with the traffic law. Another agent playing the police officer role and monitoring the traffic signal can apply a penalty (a concept related to the traffic signal concept) case the car driver are not compliant with the law. The penalty concept is used to intimidate norms violations.

UTSS Agent's Roles Modeled with the MESSAGE Methodology

Roles are modeled in most of the MESSAGE diagrams from the analysis phase. In this paper, we will only illustrate the use of the acquaintance relationships diagram, the agent/role diagram and the workflow diagram due to space limit.

The acquaintance relationships diagram, illustrated in Fig.4, shows the entities from the system and the

relationships among them. In this diagram, the roles called *Police Officer*, *Car Driver* and *Pedestrian* are represented with their acquaintance relationships. Each role knows others roles, as illustrated by the acquaintance relationships symbols. Agents playing their roles can access the ontology resource and also police or perceive/react the traffic signs according to specific relationships. Agents playing the police officer role police traffic signs while others playing the pedestrian or car driver role perceive and react to them.

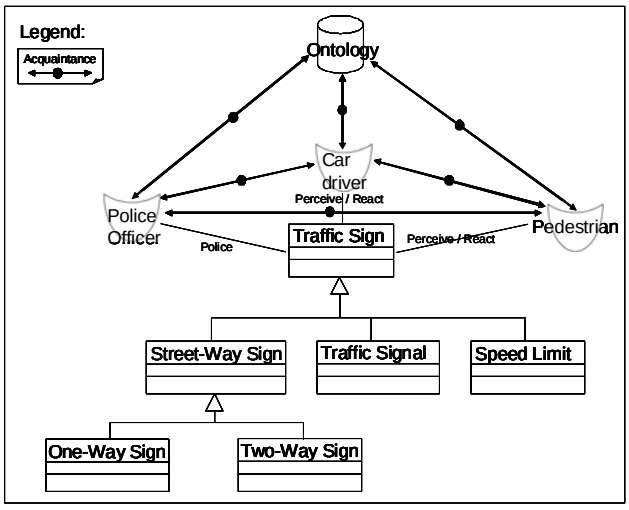

Fig.4. An acquaintance relationships diagram

The agent/role diagram focuses on the individual agents and roles, but doesn't provide how to relate goals to its specific role. The only way to do that is to build one diagram for the pair role and its goals.

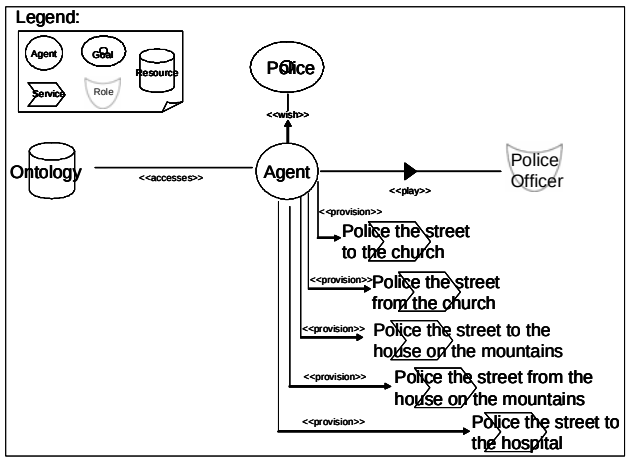

Fig.5. An agent/role diagram

Fig.5 illustrates an example where the police officer role is related to its generic goal called *Police*. This goal is specialized into the sub-goals called: *Police the street to the church, Police the street from the church, Police the street to the house on the mountains, Police the street from the house on the mountains* and *Police the street to the hospital*, related to the five services that have the same name (also illustrated in Fig.5). The ontology resource is accessed by the agents that want to know the traffic signs meanings.

The workflow diagram shows goals, tasks and the dependencies among them. Fig.6 illustrates the workflow needed to perfom the police task. Agents playing the police officer role ask for a desirable traffic sign's meaning. The ontology identify the requested traffic sign, get its meaning and, then, answer it to whom requested the information. The police task finnishs when the policed traffic sign has no more car drivers close to it.

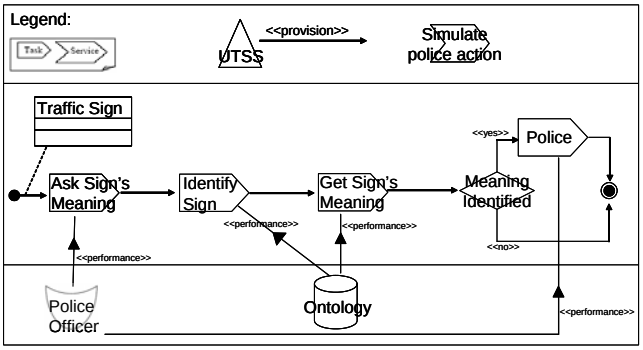

Fig.6. A Workflow diagram

To permit resources perform tasks, the workflow diagram illustrated in Fig.6 was extended. In that way, the ontology resource can be responsible to perform the "Identify Sign" and the "Get Sign's Meaning" tasks.

All case study roles modeled with the MESSAGE methodology are played by agents. By using the MESSAGE diagrams, it was possible to model the interactions between the roles, between the roles and the ontology resource, and between the roles and the traffic signs, as well as the services provided by those roles.

UTSS Agent's Roles Implemented with the ASF Framework

To implement agent roles in ASF, the abstract class *AgentRole* was extended to represent the police officer, car driver and pedestrian roles. The abstract class *Protocol* was also extended to permit the exchange of messages. The two extended classes were instantiated. After that, instances of the concrete classes *Right*, *Duty*, *CompositeBelief*, *LeafBelief*, *CompositeGoal*, *LeafGoal* and *Message* were created and associated to the instance created of the agent role extended class.

The ASF structure for the agent role class suggests that this class must have specified in its constructor its goals, beliefs, rights, duties and messages. For instance, the police officer role class, that has its services illustrated in Fig.5, has in its constructor the following goals with their names, pairs of types and values, and priorities:

- policeTheStreetToTheChurch = new LeafGoal ("policeTheStreetToTheChurch", "boolean", "true", 1);
- policeTheStreetFromTheChurch = new LeafGoal ("policeTheStreetFromTheChurch", "boolean", "true", 2);

– policeTheStreetToTheHouseOnTheMountains = new LeafGoal ("policeTheStreetToTheHouseOnTheMountains", "boolean", "true", 3);

– policeTheStreetFromTheHouseOnTheMountains = new LeafGoal ("policeTheStreetFromTheHouseOnTheMountains", "boolean", "true", 4);

– policeTheStreetToTheHospital = new LeafGoal ("policeTheStreetToTheHospital", "boolean", "true", 5);

To police all the urban places' traffic signs from the proposed environment, at least five agents playing the police officer role are required (one for each goal to be achieved).

For agents play roles and achieve their goals, plans were created in agent classes. Because agents' plans are specific to roles' goals, different types of agent classes were created, extending the abstract class *Agent* from ASF [Silva et al., 2004b]. An agent plan is related to one goal and can call various agent actions. For example, instances of agents from the *CarDriverAgent* class play the *CarDriver* role and have plans to achieve the *Drive* goal performing *Drive* actions; instances of agents from the *PoliceOfficerAgent* class play the *PoliceOfficer* role and have plans to achieve the *Police* goal performing *Police* actions; instances of agents from the *PedestrianAgent* class play the *Pedestrian* role and have plans to achieve the *Walk* goal performing *Walk* actions. Fig.7 illustrates part of the UTSS implementation tree, according to the ASF structure, with the *agent* package with its sub-packages *action* and *plan*, and the *agentRole* package.

Fig.7. The implementation by using the ASF

Agents can change roles. An agent playing the police officer role can also play the car driver role (and vice-versa) or the pedestrian role (and vice-versa); an agent playing the car driver role can change its role to play the pedestrian role (and vice-versa). To permit an easy change of roles, ASF implements all agents with one argument from the collection type that holds the roles been played by its agents.

Conclusion

This paper describes an overview of a MAS's implementation by using the role concept as a first-order abstraction. The case study was based on roles modeled with an agent-oriented methodology called MESSAGE and it was implemented by using an agent-oriented framework called Agent Society Framework (ASF).

The MESSAGE methodology successfully addresses the analysis phase producing a good result for the understanding of the problem. However, the design phase was not fully supported by the methodology. Because of that, package, class and sequence UML diagrams were used during the case study design phase.

The ASF framework facilitates the implementation phase because it suggests a structure for the solution with specific packages for agents, agent roles, actions, plans, etc., (entities modeled with the MESSAGE methodology) and has implemented classes and methods.

The implementation of the case study is an ongoing work but, the level of difficulty in the design of organizations based on roles, already can be perceived.

References

Bellifemine, F.; Poggi, A.; Rimassa, G. *Developing multiagent systems with JADE*. Proc. Intelligent Agents VII: 7th International Workshop on Agent Theories, Architectures, and Languages (ATAL), number 1986 in Lecture Notes in Computer Science, pp 89-103, Springer, Heidelberg, 2001.

Caire, G; Chainho, F; Evans, R. (2002), *Agent-oriented analysis using Message/UML*, In Agent-Oriented Software Engineering, Wooldridge, M., Weiss, G. and Ciancarini, P., Eds., Second International Workshop, AOSE 2001, LNCS 2222 Springer, Montreal, Canada, pp. 119-135.

Gruber, T. R. *A translation approach to portable ontology specifications*. Knowledge Acquisition, Vol. 5, Issue 2 (1993, June), pages: 199-220, ISSN:1042-8143.

McGuinness, D. L.; Harmelen, F. van. *OWL Web Ontology Language Overview*. W3C Recommendation 10 February 2004. In <http://www.w3.org/TR/owl-features/>. Accessed in May, 2005.

Mylopoulos, J.; Kolps, M.; Casro, J. (2001) *UML for Agent-Oriented Software Development: the Tropos Proposal*. Proc. of the Fourth International Conference on the Unified Modeling Language, Canada.

Odell, J.; Parunak, H.; Bauer, B. (2000) *Extending UML for Agents*, Wagner, G., Lesperance, Y. and Yu, E. Proceedings of the Agent-Oriented Information Systems Workshop, AOIS 2000, Eds., Austin, pp. 3-17.

Silva, V.; Lucena, C. *From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language*. In: Sycara, K., Wooldridge, M. (Edts.), Journal of Autonomous

Agents and Multi-Agent Systems, Kluwer Academic Publishers, ISSN 1387-2532, pp. 145-189, v. 9, issue 1-2, July - September, 2004a.(MAS-ML).

Silva, V.; Cortês, M.; Lucena, C. *An Object-Oriented Framework for Implementing Agent Societies*, MCC32/04. Technical Report, Departamento de Informática, PUC-Rio. Rio de Janeiro, Brazil, 2004b.

Smith, M. K.; Welty, C.; McGuinness, D. L. *OWL Web Ontology Language Guide*. W3C Recommendation 10 February 2004. In <http://www.w3.org/TR/owl-guide/> . Accessed in May, 2005.

UML. *Unified Modeling Language Specification*, version 2.0, OMG. Available at <http://www.uml.org/>. Accessed in: May, 2005

Wooldridge, M.; Jennings, N.; Kinny D. (2000). *The Gaia methodology for agent-oriented analysis and design*. Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Press, v.3, pp.285:312.

We gratefully acknowledge the financial support provided by the CNPq as part of individual grants and of the ESSMA project (552068/2002-0).