RoMAS: A ROLE-BASED MODELING METHOD FOR MULTI-AGENT SYSTEM*

QI YAN, LI-JUN SHAN, XIN-JUN MAO AND ZHI-CHANG QI

Dept. of Computer Science and Technology, National Univ. of Defense Technology
ChangSha, 410073, China
E-mail: yanqi_nudt@yahoo.com.cn

This paper proposes a new definition for role, and estimates role's functionality in MAS (Multi-Agent System) analysis, design and implementation. A set of graphical notations and a method RoMAS (Role-based Modeling for MAS) are presented. A case about RoboCup simulation is studied to illustrate the applying of the modeling language.

1. Introduction

Along with the progress of computer network and communication, the research on MAS has become one of the hotspots in DAI [1,2]. Meanwhile, the concept of role and role model [3,4,5,6], as a facility of abstraction and decomposition, has attracted the Agent-Orientation and Object-Orientation researchers.

We have investigated some famous works, including the Gaia [9] methodology, UML [10] and OOram [8] (the latter gives a foundation of role concept to UML). In this paper, we present a new concept of role, clarify its meaning and properties, analyze its significance in MAS (Multi-Agent System) and propose a method RoMAS to realize its potential. Our method supports dynamic binding between role and agent.

The remainder of the paper is organized as follows. Section 2 gives a new definition for role different from the existing ones; Section 3 overviews the related works and their shortcomings; section 4 gives a definition of role organization and illuminates its significance for MAS development; section 5 proposes a diagrammatic modeling language and demonstrate it by a case; finally the future works are discussed.

2. Role in MAS

Our motivation is to model MAS based on role. Several MAS methodologies

* This work is supported by the National Natural Science Foundation of China under Grant No.600003002; the National Grand Fundamental Research 973 Program of China under Grant No.G1999032700; the National High Technology Development 863 Program of China under Grant No.2002AA116070.

have adopted the concept of role (or role model) in analysis and design phases, such as MaSE [7] and Gaia. In Gaia, a role is *"defined by four attributes: responsibilities, permissions, activities, and protocols*." Thus it is an implicit definition.

Our insight is that an entity (i.e. object in OO or agent in MAS) cannot exist by itself, and it should bind some roles. We propose a definition of role in MAS:

Role: (1) From the conception perspective, a role is a constraint under which an agent takes part in some interactions and evolves in a certain way. In MAS, an agent behaves under its bound roles.

(2) From the implementation perspective, a role is an encapsulation of certain attributes and behaviors of the agent it is bound to.

The characteristics of this definition are: 1) An agent can have more than one role at one time; 2) An agent can dynamically change its roles; 3) Role does not take actions while it is the agent that takes actions; 4) Roles are not isolated: there must be other roles related to it; 5) Role acts as a "window" of agent, through which other agents know the way to interact with the agent; 6) Roles provide a facility for efficient reuse.

3. Related Works

Role concept has long been concerned since the advent of Object-Orientation, within which the works [8,9] are famous. The researches have revealed some of its features, such as carrier-independence, dynamics, etc.

Concerning role's effect in MAS development, in the Gaia and MaSE methodology, roles are the result of analysis as undertakers of system goal, and turned into agent classes in the implementation phase, while the concept of role per se disappears. Besides, because of the fixation of role to agent, the interactions among agents are decided according to the system goal in the phase of design. As a result, an agent cannot change its roles at run-time.

4. Role Organization in MAS

Roles along with the communication paths among them compose a role organization. Role organization modeling addresses software specification, analysis, and design. Many of them are documented patterns. Role organization is helpful in the following aspects:

(1) It represents how agents interact. Each agent acts and communicates under its role or roles.
(2) It abstracts interactions in MAS.
(3) As a frame in which agents bind certain roles, it enables the agents to change their roles dynamically.
(4) Essentially, role organization is an ad hoc role [14]. It can be

instantiated, generalized, specialized, and aggregated.

Take the example of a football team: the typical roles (goalkeeper, linebacker, vanguard and attacker) compose a role organization (Figure 1). Note that except goalkeeper, specializations of each role (for example, left-backer, right-backer and middle-backer as specializations of linebacker) compose role organizations as well.

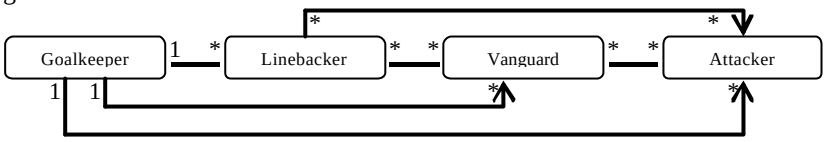

Fig.1 the pattern of football team roles

The rectangle in figure 1 denotes role, and notation "1" or "*" near link ends denotes quantities of the role.

5. Role-Based MAS Development

We propose a role-based modeling language tailored to (1) explicitly separate role from agent conceptually and linguistically; (2) roles exist throughout the whole process of MAS development. The main development process in natural language is as follows:

(1) Captures use cases;
(2) Identify roles from use cases;
(3) Constructs role organization;
(4) For each role, if the appropriate agent does not exist, then go to (5); else
    I. Binds roles to agents
    II. Describes dynamic properties of bind relation between agents and roles
    III. Go to (6)
(5) Generates agents according to roles; Go to (4).I.
(6) Generates codes for agents with roles bound;

Later discussion will be concentrated on RobCup simulation football team.

5.1. *Capture Use Cases*

Use cases outline the system events and their interactions. We adopt Use Case Diagram to describe use cases. Figure 2 represents a use case (there should be

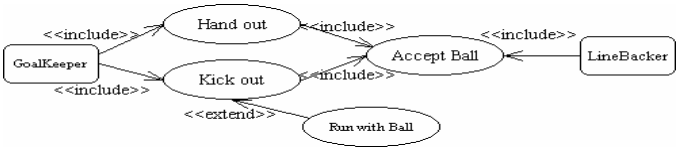

Fig.2 Use Case example

more in real system), in which rectangle denotes Actor and ellipse denotes Use Case. In this example, *Goalkeeper* actor includes *Hand Out* use case or *Kick Out* use case to pass the football; *Linebacker* actor includes *Accept Ball* use case to

get the football. In some possible conditions, *Kick Out* use case may be extended by *Run With Ball* use case. See [10] for detailed information about <<include>> and <<extend>> stereotypes.

5.2. *Identify Roles*

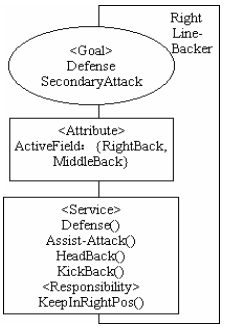

Fig.3 Role example -Right Linebacker

Roles can be identified from use cases [11,13]. However, use cases are not sufficient for describing all the roles and events in the MAS. An assistant method is to check the words with *–er*, *-ist* or *–or* suffix in the requirement specification. Figure 3 shows an example notation of role, in which the right top text is the role's name, the ellipse with text shows the goals the role takes, the middle rectangle with text shows the attributes, the bottom rectangle with text shows the services the role provides and its responsibility. For detailed information about goal, service, responsibility, readers are referred to [14].

5.3. *Construct Role Organizations*

Roles are not isolate. Every role communicates and interacts with other roles. Besides, roles can be specialized or aggregate to other roles. Inheritance and aggregation associations respectively denote specialize/generalize and aggregate/decompose relations among roles. Figure 4 shows a role organization, in which rectangle with a semi-circle denotes role; arrow line denotes communication path, triangle denotes inheritance relation, diamond denotes aggregation relation, and rectangle with a line on left-top corner denotes organization. For semantics of the notations, readers are referred to [3].

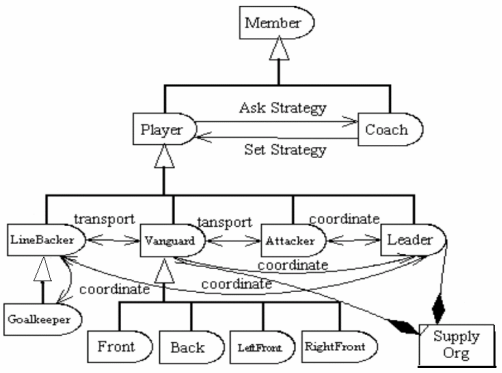

Fig.4 Role Organization example- team

5.4. *Bind Roles to Agents*

For each role, the appropriate agent may exist or not. For existing agents, they are classified to agent classes directly (figure 5). An agent has a name, attributes,

actions and behavior rules. Inexistent agents are related with roles and thus can be generated from roles. For detailed methods, readers are referred to [14].

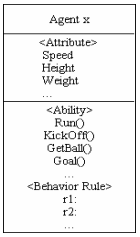

Fig.5 Agent example

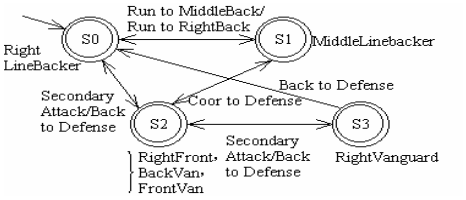

Fig.6 Role transition example

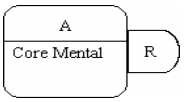

Fig.7 Role Binding

An agent can change its roles dynamically. To make this property clear, we apply finite automata to describe agent's role transitions (Figure 6). Circle represents state, which denotes the roles bound to the agent. Arrow line denotes state transition, which is executed under the conditions or messages listed near the line. The notation " }" / " {" denote OR / AND relation of the roles indicated on its right-side. For " }" , only one of the roles is bound to the agent; for " }" , all the roles are bound to the agent.

Figure 7 describe initial binding of role to agent. The rectangle is a compact form of agent and the rectangle with semi-circle is a compact form of role.

6. Conclusions and Future Work

Roles represent system goal and constrain agents' behavior. They exist throughout all phases from analysis to implementation, and this is the main difference to previous works on role both in OO and in AO. We believe such a model can enable a natural realization of dynamic bindings between agents and roles.Besides, the RoMAS method generates roles from use cases. This prevents jumping from abstract use cases to concrete entities.

Further work focuses on designing and developing an application with RoMAS: 1) By introducing the concept of role and organization to the system, enable every agent in the system to take on roles according to its own mental state and its situation, so as to improve the system's adaptive and collaborative ability. 2) Based on the study of role concept in MAS and further analysis of its dynamics, take the RoboCup as a case, design and develop a simulation client to testify RoMAS' pragmatic efficiency.

Acknowledgments

Thanks to Professor Hong Zhu and Miss Xin Wang for their heuristic discussion.

References

1. M. Wooldridge and Paolo Ciancarini, *Agent-oriented software engineering: The state of the art*, in Handbook of Software Engineering and Knowledge Engineering. 2001, World Scientific Publishing.
2. Sycara, K. 1998. *Multiagent systems*. AAAI AI Magazine 19(2).
3. Andersen, E. (Egil), *Conceptual Modeling of Objects: A Role Modeling Approach*, PhD Thesis, University of Oslo, 1997. ftp.nr.no/pub/egil/ConceptualModellingOO.ps.gz
4. KenDall, E. A., *Agent Roles and Role Models: New Abstractions for Multiagent System Analysis and Design*, International Workshop on Intelligent Agents in Information and Process Management, Germany, September, 1998
5. Richle, D., T. Gross, *Role Model Based Framework Design and Intergration*, OOPSLA'98, Proceedings of the 1998 Conference on Object-oriented Programming Systems, Languages and Applications, ACM Press, 1998.
6. Richle, D., Bureaucracy, *in Pattern Languages of Program Design 3*, R. Martin, D. Richle, F. Buschmann (Ed.), Addison Wesley, 1998, pp. 163-185
7. S. A. DeLoach. *Analysis and design using MaSE and agentTool*. In Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001), 2001.
8. Reenskaug, T., Wold, P., Lehne, O. A., *Working with Objects, The OOram Software Engineering Method*, Manning Publications Co, Greenwich, 1996.
9. M. Wooldridge, N.R. Jennings, and D. Kinny. *The Gaia methodology for agent-oriented analysis and design*. Journal of Autonomous Agents and Multi-Agent Systems, to appear 2000. 15.
10. UML Specification, www.omg.org/technology/documents/formal/uml.htm.
11. Jacobson, I., et al., *Object Oriented Software Engineering: A Use Case Driven Approach*, Addison Wesley, 1992.
12. Zhu, H. *SLABS: A Formal Specification Language for Agent-Based Systems*, Int. J. of Software Engineering and Knowledge Engineering 11(5) (Nov. 2001), 529-558.
13. Kendall, E. A., U. Palanivelan, S. Kalikivayi, "*Capturing and Structuring Goals: Analysis Patterns*," EuroPlop'98, European Pattern Languages of Programming, Germany, July 1998.
14. Qi Yan, Xin-Jun Mao and Zhi-Chang Qi, *Modeling role-based organization of agent system*, UKMAS'02.
15. Emiliano Tramontana, *A Reflective Architecture for Role-Based Designs*, COMPUTING SCIENCE February 2000.