Modelling Distributed Agent Systems with RoMAS

Qi Yan, XinJun Mao, and ZhiChang Qi

Department of Computer Science and Technology,
National University of Defense Technology,
Changsha, 410073 China
`yanqi_nudt@yahoo.com.cn`

Abstract. Multi-agent systems(MAS) are typical distributed systems. To model such a distributed system, a new definition and architecture for agent is proposed to be more appropriate for dynamic, open system's analysis, design and implementation. The formal specification of these concepts is described. A MAS modelling methodology RoMAS based on these definitions is presented through RoboCup simulation football team case, including its graphical modelling language and the complete process.

1 Introduction

Agent-Oriented Software Engineering (AOSE), Multi-Agent System (MAS) [1, 2] are hot spots in distributed systems research and development field. Meanwhile, Agent-Based Social Simulation (ABSS) receives social researchers' great attention, e.g. RoboCup [3] has already become an ideal test-bed for both ABSS and MAS, which are typically distributed systems.

Since it is agent that associates ABSS and MAS, the first question we need to answer is "what are agents and multi agents system?". This paper will address this question in a formal way.

The remainder of this paper is organized as follows. Section 2 illustrates the homogeneousness between human being and software agent. Section 3 presents the main idea of soft gene, role and agent, meanwhile defines corresponding concepts in Z notations. Section 4 illustrates RoMAS(Role-based MAS Modelling Methodology) using the case of RoboCup simulation football team . Section 5 discusses related works, gives some conclusions and discusses the future works.

2 From Human Study to Distributed Multi-Agent Based Simulation

Existing agent-oriented (AO) software methodologies(see [4], e.g. Gaia [5], usually take an agent as *a software component that autonomously behaves and collaborates with others in certain environments.* But how to add or change interaction

of new types between agents at run-time? These problems are cause by open and dynamic properties of a MAS and have not been addressed by the methodologies [6]. Current AO software methodologies require multiple inheritance and dynamic inheritance mechanisms to deal with the problems, while they are recognized as negative factors for system robustness, maintainability. To address these problems, we may need a new understanding about agent.

Karl Marx said in *On Feuerbach, 1845*, "*The first premise of all human history is, of course, the existence of living human individuals. Thus the first fact to be established is the physical organization of these individuals and their consequent relation to the rest of nature.*"

Because of limited space, the details on human/society are omitted here. The point is that when we homogeneously map human properties to agent (although we have not defined agent yet, we still can use this word to see what properties it should have), we get figure 1. It represents:

1. What activities does an agent do? An agent does two typical kinds of activities (noted as rectangles, the curved lines connecting circle/ellipse side and rectangles represent an association relation):(1)Practice Activity (arrows between agent and environment/itself): An agent lives in the software system, it should can apperceive from/react to the system. Such an activity should be under something's control. We call it soft genes (noted as SGi in figure 1, i is 1,2,3,etc.) which is defined in the next subsection; (2)Social Interaction Activity (arrows between agents represent social interaction activities, the curved lines connecting arrows and rectangles represent an association relation): An agent lives in a virtual society, it takes certain social roles so as to interact with others.
2. What models does an agent use? Agents use these typical kinds of models (noted as rectangles, the curved lines connecting circle/ellipse side and rectangles represent an association relation, the curved dashed lines represent some activity use some models):(1)Model on itself: Agents use it to determine what to do at certain time; (2)Role Organization Model and Communication Protocol: Agents use these models to perform social interaction activities.
3. What relations does an agent have? In 'SGi' layer (smaller ellipses), it relates to itself and it makes practice activities; in 'Role organization' layer (bigger ellipses), it relates to other agents through their roles and it makes social interaction activities; in 'Environment' layer (circles), it relates to virtual world objects and it makes practice activities.

We propose two concepts, soft gene and role, for agent. When we degenerate 'environment' to an inactive agent (that's to say, the environment is seldom affected by other system agents), the three layers agents live in can be reduced to role layer and soft gene layer. The importance of such a reduction is that we can use as least concepts as possible to simulate a society with MAS.

We need to be formal to be precise about the concepts we discuss, yet we want to remain directly connected to issues of implementation. Z provides just those qualities that are needed, and is increasingly being used for specifying frameworks and systems in Distributed Artificial Intelligence (DAI) e.g. [7].
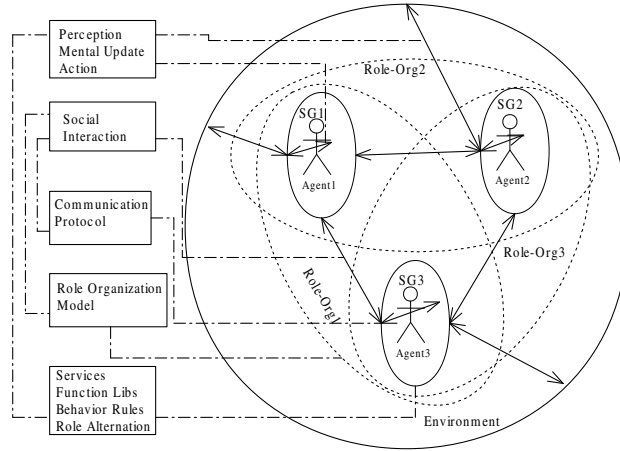
Fig. 1. Agent's activities, models and layers

3 Soft Gene, Role and Agent and Their Formal Specification

The meta-model of multi-agent system is as below: an agent is composed by a certain set of soft genes bound with a certain set of roles; an agent's ability of action is determined by its gene; an agent communicates with others through its bound roles.

Before we can move to definition of any of above concepts, we must first define some primitives.

Definition 1. *An behavior is an action's being called from outside/self and performed to cause a discrete event that changes the state of the environment.*

Each behavior must satisfy certain conditions before executing. The preconditions and post-condition(both are logic predications) together with behavior make a behavior rule, which can be represented as:

$$< pre - condition > behavior < post - condition >$$

Definition 2. *An attribute is a perceivable feature (of the world).*

3.1 Soft Gene

A football (an object in a computer simulation football match) is a software component and its soft genes are *shape, color, texture, springiness, size, weight* and so on. However, soft genes are not just attributes. Considering another software component in a simulation football match, a player, his/her soft genes are both attributes (*height, weight, run speed* and so on) and behaviors (*run(),*

goal(), grab() and so on) with the ways in that those behaviors perform (some player like to give the football to teammates while some others will take the football by themselves). Thus we get:

Definition 3. *A soft gene is an entity that comprises a set of behaviors and a set of attributes.*

In Z, before constructing a specification, we must first define types. Here we define the set of all behaviors and the set of all attributes:

[*Behavior, Attribute*]

Now a state schema can be constructed that defines a soft gene. (For the meanings of the Z notations, readers are referred [8].) In the schema, *e_cando, i_cando* are the sets of behaviors(external observable or internal observable) of the soft gene; *e_attributes, i_attributes* are the sets of features(external perceivable or internal perceivable) of the soft gene. Soft genes are therefore defined by their ability in terms of their behaviors, and their configuration in terms of their attributes.

```
┌─ Softgene ────────────────────────────
│  e_cando :  P Behavior
│  i_cando :  P Behavior
│  e_attributes :  P Attribute
│  i_attributes :  P Attribute
└───────────────────────────────────────
```

3.2   Role

A role represents the expected social behavior of an individual. All roles as a whole can be regarded as the physical organization of a multi-agent system. A role is something that satisfies a goal or set of goals, provides some services.

Definition 4. *A goal is a state of affairs to be achieved in the environment.*

Definition 5. *A service is a behavior framework, which determines how the behavior be performed by the soft gene that binds the role.*

$Goal == P\ Attribute$
$Service == P\ Behavior$

The schema for a role is simply that of some attributes with the addition of goals and services.

```
┌─ Role ────────────────────────────────
│  services :  P Service
│  goals :  P Goal
│  e_attributes :  P Attribute
│  i_attributes :  P Attribute
├───────────────────────────────────────
│  goals ≠ φ
│  services ≠ φ
└───────────────────────────────────────
```

Some properties of role can be found in [9].

3.3 Agent

Each agent holds some soft genes and takes on one or more roles.

Definition 6. *An agent is an entity that comprises some soft genes together with some bound roles.*

The relation between soft genes and roles is maybound, and the bind method is used to make such a relation between certain soft genes and roles.

$$maybound : Softgene \leftrightarrow Role$$

$$
\begin{array}{|l}
\underline{\quad Agent \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad} \\
softgenes : \ \mathbb{P} \ Softgene \\
roles : \ \mathbb{P} \ Role \\
bind : \ Role \nrightarrow Softgege \\
\hline
softgenes \neq \phi \\
roles \neq \phi \\
bind \subseteq maybound \\
\end{array}
$$

Because of limited space, we omitted other important schemas, e.g. agent state, agent act, etc. The architecture of such an agent is represented as figure 2. An agent is composed by several soft genes that bound with certain roles (e.g. role1+soft gene1, role2+soft gene2 in figure 2). Both soft genes and roles have some mental attitudes that determine how and what to do by this agent. When a message is receives by one soft gene, it should be determined that who (the soft gene or the roles that bound to it)will be responsible to the message. Either soft gene or roles may use their mental attitudes to make an appropriate plan including a sequence of actions. The actions may cause certain messages to be sent to others.

3.4 Be Intelligent to Live in Open and Dynamic Systems

In an open and dynamic distributed system, agents need to know what/how they can do by itself and what/how they can do with others. So long as an agent can alter its bound roles, we say it has the intelligence to interact with others under an open and dynamic environment for reasons below:

1. The set of soft genes in an application is quite stable. Every individual (without interactions with others) is composed by some genes.
2. An agent can bind or discard some roles at run-time. This character is particularly useful for open and dynamic systems:
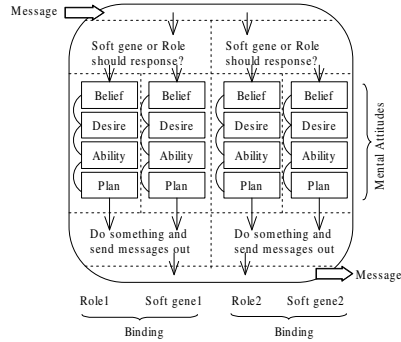
Fig. 2. The architecture of agent

    (a) To create roles of some new types, the system just reads new role specifications and create corresponding roles.
    (b) After discarding some old members, the system organizes interactions among remaining agents through their bound roles.
    (c) By binding or discarding roles, an agent may dynamically alter its way of interaction with other agents.
3. For implementation, multiple inheritance will be not necessary and then corresponding side effects can be prevented.

As we have discussed in section 2, other agent definitions and methodologies can not address these problems well, such as Gaia [5] or MaSE [10, 6].

4 RoMAS: Role-Based MAS Modelling Methodology

Based on the definitions we have given, we propose a role-based modelling language and methodology tailored to (1) explicitly separate soft gene and role from agent conceptually and linguistically; (2) roles exist throughout the whole process of MAS development. To demonstrate our language, we take the case of RoboCup [3]football team simulation. The main development processes in natural language is as follows:

1. Capture use cases;
2. Identify roles from use cases;
3. Construct role organization;
4. For system components, distill their soft genes;
5. For each role, considering soft genes, if the appropriate genes does not exist, then go to 6; else
    (a) Binds roles to genes and generate agents
    (b) Describes dynamic properties of bind relation between agents and roles
    (c) Go to 6
6. Generates agents according to roles; Go to 5.(a).
7. Generates codes for agents with roles bound;

4.1 Capture Use Cases

Use cases outline the system events and their interactions. We adopt Use Case Diagram to describe use cases. Figure 3 represents a use case (there should be 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 [11]for detailed information about $<< include >>$ and $<< extend >>$ stereotypes.
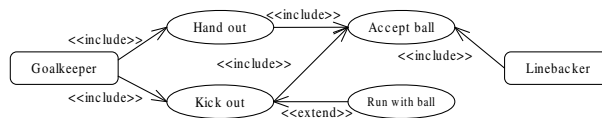

Fig. 3. Example of a use case of football team

4.2 Identify Roles

Roles can be identified from use cases [12, 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 4 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 more information about these concepts, readers are referred to [14].
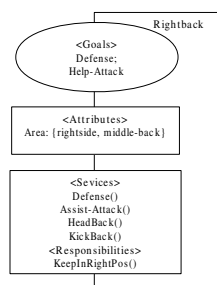

Fig. 4. Example of a role in a football team

4.3 Construct Role Organizations

Roles are not isolate. Every role communicates and interacts with other roles to fulfill its goals. Besides, roles can be specialized or aggregate to other roles. Inheritance and aggregation associations respectively denote the specialize/generalize and aggregate/decompose relations among roles. Figure 5 shows an example of 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 [15].
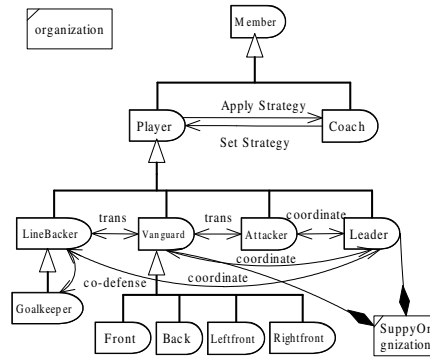

Fig. 5. Example of a role-organization in a football team

4.4 Distill Soft Genes

Check system individuals and distill their genes. These soft genes later will be inherited to become other genes. The genes determine agent's attributes and behavior with rules.

4.5 Bind Roles to Soft Genes and Generate Agents

For each role, the appropriate soft genes may exist or not. For genes (instances) in existence, they (with the roles) are classified to agent directly. For inexistent soft genes, our insight is that they are related with roles and thus can be generated from roles. For detailed methods, readers are referred to [14].

An agent can change its roles dynamically. To make this property clear, we apply finite automata to describe agent's role transitions. See the example in 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.
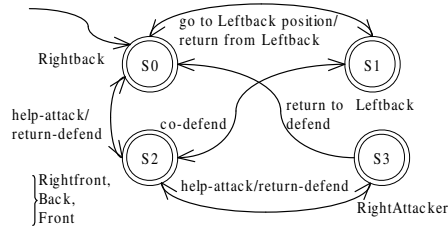
Fig. 6. Example of role transitions of football team

5 Related Works and Conclusions

Role concept has long been concerned since the advent of OO (Object-Oriented), a famous work is [16]. Concerning role's effect in MAS development, Gaia and MaSE methodologies think 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.

In this paper, we propose a new definition of agent based on soft gene and role. The new definition leads us to a diagrammatic role-based modelling method supporting MAS analysis and design. With RoMAS, the ability and behavior embodied by an agent in MAS is a combination of its soft genes and its roles. Soft genes state the basic perceptive and behavior abilities; roles represent system goal and constrain agents' behavior. They exist throughout all phases from analysis to implementation, which enables a natural realization of dynamic bindings between agents and roles.

Further works focus on developing an application with RoMAS by introducing the concept of soft agent, role and organization to the system, thus enabling 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.

Acknowledgments

References

1. Jennings, N.R.: On agent-based software engineering. Artificial Intelligence **117(2)** (2000)
2. Wooldridge, M., Ciancarini, P.: Agent-Oriented Software Engineering: The State of The Art . in Handbook of Software Engineering and Knowledge Engineering (2001,World Scientific Publishing)

3. RoboCup: (http://www.robocup.org)
4. Iglesias, C., Garijo, M., Gonzalez, J.: A Survey of Agent-Oriented Methodologies. Intelligent Agents V (1999)
5. Wooldridge, M., Jennings, N., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems **3** (2000) 285–312
6. Deloach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent Systems Engineering. International Journal of Software engineering and Knowledge Engineering **11(3)** (2001)
7. Luck, M., d'Inverno, M.: A formal framework for agency and autonomy. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95) (1995) 254–260
8. Spivey, J.M.: The Z Notation: A Reference Manual. Prentice-Hall (1989)
9. Yan, Q., Shan, L.J., Mao, X.J.: RoMAS: A Role-Based Modeling Method for Multi-Agent System. Proceedings of International Conference on Active Media Technology 2003 (to be pressed)
10. DeLoach, S., Wood, M.: Developing multiagent systems with agenttool. Intelligent Agents VI **1757** (2000)
11. OMG: http://www.omg.org/technology/documents/formal/. Specification of UML (1995)
12. Jacobson, I.: Object Oriented Software Engineering: A Use Case Driven Approach. Addison Wesley (1992)
13. KenDall, E.A., Palanivelan, U., Kalikivayi, S.: Capturing and Structuring Goals: Analysis Patterns. EuroPlop'98, European Pattern Languages of Programming (July 1998)
14. Yan, Q., Mao, X.J., Qi, Z.C.: Modeling Role-Based Organization of Agent System. UKMAS'02 (2002)
15. Andersen, E.E.: Conceptual Modeling of Objects: A Role Modeling Approach. PhD thesis, PhD Thesis, University of Oslo (1997)
16. Reenskaug, T., Wold, P., Lehne, O.A.: Working with Objects, The OOram Software Engineering Method . Manning Publications Co,, Greenwich (1996)