

Modelling Soft Gene, Role and Agent in Extended UML

Qi Yan and others

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

Abstract. With the recent rapid growth of interest in Multi-Agent Systems(MAS), has come an associated difficulty modelling basic terms, concepts. In this paper, the underlying homogeneousness between human being (society) and software agent (MAS, Multi-Agent System) is opened out, based on which a new definition and architecture for agent is proposed. An agent is made of certain soft genes and roles. Such agents can be more appropriate for dynamic, open system's analysis, design and implementation. The associated structure model and behavior model are described in extended UML.

1 Introduction

Agent-Oriented Software Engineering (AOSE), Multi-Agent System (MAS) [1, 2] and Agent-Oriented Programming (AOP) [3, 4] are hot spots in software development field. They are even regarded as a revolution by some researchers [5]. Meanwhile, Agent-Based Social Simulation (ABSS) receives social researchers' great attention, e.g. RoboCup [6] has already become an ideal test-bed for both ABSS and MAS.

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 and propose a MAS modelling methodology to make multi-agent based social simulation (MABS).

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. Section 4 models agents structure and behavior. Section 5 illustrates the usefulness of the concepts by modelling a RoboCup [6] simulation football team using RoMAS (Role based MAS modelling methodology) [7]. Section 6 discusses related works. Section 7 gives some conclusions and discusses the future works.

2 From Human Study to MABS

From Descartes, Kant, Floyd, etc. to Karl Marx, after their studying of nearly 400 years, human essence research recognized that human individual holds both

natural and social properties [8]. Such a notable theory fruit may help software researchers to understand agent and multi-agent system for that human and society give good example for agent and multi-agent system, from basic apperceive, interaction to challengeable dynamic, open properties.

2.1 Traditional Agents: Make Actions and Interactions

Current agent-oriented methodologies (see [9] for a survey of them) are lack of the ability to deal with distributed systems' open, dynamic properties for which the radical reason is the understanding and definition of *agent*.

Existing agent-oriented (AO) software methodologies, e.g. Gaia [4], 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 [10]. 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. We notice that an agent is quite like a human individual in one society — it can make actions and interact with others, which gives us big intellectual enlightenment.

2.2 Human Individuals Have Genes and Roles

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.*"

As Karl Marx mentioned, '*the existence of living human individuals*' and '*the physical organization of these individuals*' are essential to any intelligent individual. Our understanding is that two concepts are essential to creatures: gene and role. A gene is a hereditary unit that occupies a specific location on a chromosome and determines a particular characteristic in an organism. Genes exist in a number of different forms and can undergo mutation [11]. A role is the characteristic and expected social behavior of an individual [11].

Figure 1 shows an example of genetic code. A sequence of three adjacent nucleotides constituting the genetic code that specifies the insertion of an amino acid in a specific structural position in a polypeptide chain during protein synthesis.

2.3 New Agents: Learn from Human through Homogeneous

To understand human beings, several questions need to be answered. They are described in figure 2. It represents:

		2nd base in codon					
		U	C	A	G		
1st base in codon	U	Phe Phe Leu Leu	Ser Ser Ser Ser	Tyr Tyr STOP STOP	Cys Cys STOP Trp	U C A G	
	C	Leu Leu Leu Leu	Pro Pro Pro Pro	His His Gln Gln	Arg Arg Arg Arg	U C A G	
	A	Ile Ile Ile Met	Thr Thr Thr Thr	Asn Asn Lys Lys	Ser Ser Arg Arg	U C A G	
	G	Val Val Val Val	Ala Ala Ala Ala	Asp Asp Glu Glu	Gly Gly Gly Gly	U C A G	

Fig. 1. An example of genetic code [12]

1. What activities does a human do? Human beings do two typical kinds of activities (noted as rectangles):(1)Practice Activity: A human lives in the world, he/she can apperceive from/react to the world, E.g. when a football player kicks a football, he/she is doing a practice activity. Such an activity may be the reaction to environment's alternation, instinctive actions under genes' control, etc.; (2)Social Interaction Activity: A human lives in a society, he/she takes certain social roles so as to interact with others, E.g. when a football player (as a vanguard) kicks off a football to his/her teammates (maybe an attacker), a social interaction activity is happening.
2. What medias does a human use? Human beings use two typical kinds of medias (noted as rectangles):(1)Tool System: Human beings use tool system to perform practice activities (as the dashed line describes), e.g. eyes, ears, mouth, hammer, etc.; (2)Social Interaction System: Human beings use social interaction systems to perform social interaction activities, e.g. English grammar, contract, etc..
3. What relations does a human have? Three circles respectively represent three layers in which an agent relates to itself, other agents or the environment. The 'Being'is the body of a human, before it is combined with some 'Individual' and to 'Role organization', it can not do anything. In 'Individual' layer, it relates to itself and it makes practice activities (as the dashed line describes); in 'Role organization' layer, it relates to other agents through their roles and it makes social interaction activities; in 'Environment' layer, it relates to real/virtual world objects and it makes practice activities.

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 3. 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, E.g. when a software

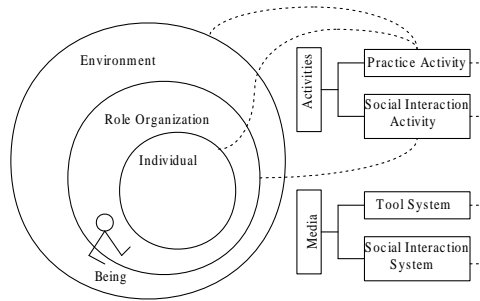


Fig. 2. Human being's activity, media and layers

football player kicks a football, it is doing a practice activity. Why and how such activities be performed? Such an activity should be under something's control. We call it soft genes (noted as SGi in figure 3, 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, E.g. when a software football player (as a vanguard) kicks off a football to its teammates (maybe an attacker), a social interaction activity is happening.

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 Function Libs, Behavior Rules, Role Alternation Rules (see section 4) 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.

Since that we have already complete the homogeneousness, now it is necessary for us to distill substantial concepts from the complexity as figure 3 shows.

2.4 A New Understanding for Agents

Thinking from a bionic point of view, software agents bear an analogy to human beings for that they both require the properties of autonomy, reactive, etc. We propose two concepts, soft gene and role, for agent.

1. A soft gene (for agent) is a hereditary unit that determines some particular characteristics in a software component. After adopting certain soft genes,

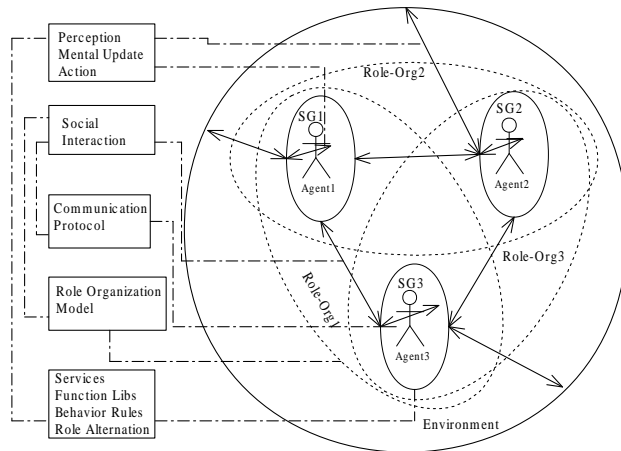


Fig. 3. Agent's activity, model and layers

the software component can make certain practice activities, update its own behavior rules, etc. In other words, a software component with certain soft genes can do something and knows how to do them by itself.

2. A role (for agent) is the characteristic and expected social behavior of a software component. After binding certain roles, the software component can know and make social interaction activities. In other words, a software component with certain roles can do something and knows how/why to do them with others.
3. An agent is *a software component that composed by certain soft genes and bound with certain roles.*

Figure 4 illustrates the meta-model.

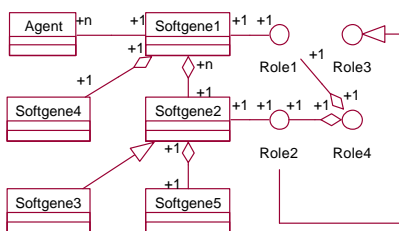


Fig. 4. The meta model in UML

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.

3 Soft Gene, Role and Agent

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.

3.1 Soft Gene

We have defined a soft gene is a hereditary unit that determines some particular characteristics in a software component. For example, 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. If we define a component with such genes:

< Genes -- shape : round; color : black and white; size : 24cm; texture : leather; weight : 430g; springiness : good >

It can almost be assured that we are talking about a football. 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). Figure 5 is an soft gene example in UML, which includes two soft genes of football players: static one defines player's basic body attributes and abilities while the behavior one defines player's countermoves attributes and abilities.

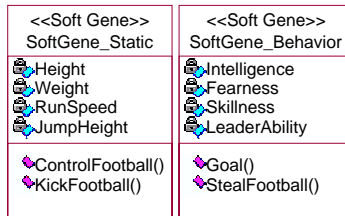


Fig. 5. The example of soft genes in UML

Once the soft genes are determined, the software component's 'existence of living ... individuals' is completely described. ('The physical organization of these individuals' will be discussed in the next subsection.)

Example 1. Code of a soft gene. Method represents the 'behaviors' we mentioned above. The *WithBall* is pre-condition and post-condition (both are logic predication), which describe behavior rules.

```

Gene SoftGene_Behavior {
  Attribute:
    double Skillness;
    double Fearness;
    double LeaderAbility;
    double Intelligence;
  Method:
    <!WithBall >StealFootball(Target); < WithBall >
    < WithBall >Goal();
}

```

Some additional properties of soft gene should be illustrated:

1. Heteromorphism: soft genes can be organized into different granularities.
 - (a) As software production: the genes are 1 and 0. Every software production is a sequence of 0 and 1;
 - (b) As model: the genes are chosen characteristics of modelling entities. For example, when we simulate a football player, we just need to consider his/her weight, height, characteristics, ability of kicking not painting;
 - (c) In implementation: the genes are certain programming language mechanisms, such as *Class*, *Interface* in C++, Java and so on.
2. Inhomogeneity: different applications require different soft genes. To simulate an office system, we care about officers' name, salary, ability to use computer, etc.; to simulate a football player, we care about his/her weight, height, characteristics, ability of kicking not painting.
3. Inheritance: genes can inherits from other genes. It is important for reuse.
4. Aggregation: genes can be aggregated into a bigger gene.

3.2 Role

Soft genes determine software individuals' existence of living, '*The physical organization of these individuals*' makes the backbone of one system. 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. For the last example, if a player has such role description:

< Roles : – – Always : Left guard; Possibly : Left front >

We can state that this player should defense attacks from left side; most time he/she communicates with the goal keeper role. Figure 6 is an example of role *GoalKeeper* in UML. It is an interface that has corresponding attributes and methods.

Some relations between roles are:

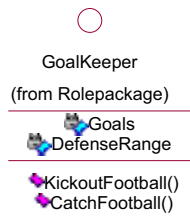


Fig. 6. The example of role in UML

1. Relational: In any system, a role almost always appears with at least another role that it interacts with. E.g., whenever there is a role of teacher, there must be a role of student. Otherwise, the role of teacher is meaningless. In summary, a role defines the relationships among individuals, also the interface, protocol and functionality of the interactions among individuals.
2. Hierarchical: Those individuals who play a role in an organization can be highly organized. E.g., the lecturers in a university's department can be further divided into subject groups and play roles like lecturer in computer science and lecturer of information systems, etc. In this way, the duty and tasks of a role at a higher level are fulfilled through interactions between roles at a lower level. Therefore, a role organization can be decomposed hierarchically.
3. Stable: The representation of the structure of a dynamic system in the form of a role organization is usually stable in the sense that the relationships between the roles do not change so frequently as the those between the individuals of the system. However, the relationships between the roles are not always constant. In human organizations, new role can be formed and existing roles can be deleted or merged into another role, etc. The revision of the role organization by the organization itself leads to self-organized.

Example 2. Code of a role. Goal is used in RoMAS [7] Attributes and Method describe role's states and abilities, etc.

```

Role r1 {
    Goal: logic formulae
    Attributes: similar to gene's
    Method: similar to gene's
}

```

Roles altogether with the interaction paths among them compose a role organization. Many of role organizations are documented patterns (see Figure 7). Role organization is helpful in:

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.

- Essentially, roles can be organized into another role [13]. It can be instantiated, generalized, specialized, and aggregated.

Considering a football team: the typical roles (goalkeeper, linebacker, vanguard and attacker) compose a role organization (Figure 7). Note that except goal-keeper, specializations of each role (e.g. left-backer, right-backer and middle-backer as specializations of linebacker) compose role organizations as well.

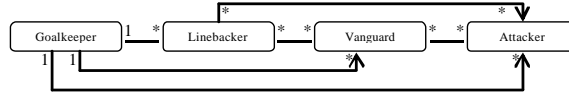


Fig. 7. The pattern of football team roles

The rectangle in figure 7 denotes role; the links represent communication paths; and notation 1 or * near link ends denotes quantities of the role.

In comparison with existing works in which role and role organization are utilized in analysis and design, we believe that as facilities for MAS modelling, they should also be applied in implementation process. The reason will be illuminated in section 6: *Related Works*.

3.3 Agent

Each agent holds some soft genes and takes on one or more roles. The soft genes realize some roles so as to make an agent. Figure 8 is an example of agent in UML. The relationship between soft genes and agent is *compose* that means the soft genes compose the agent; the relationship between roles and soft genes means that the soft genes *play* the roles. Notice that the relation between soft genes and agent can be grouped into visible and invisible, which have been used in the example 3, and while the meaning has been illustrated by notes, the complete semantics will have to be given in other papers because of limited space.

Figure 8 shows that soft genes (Sg_1, Sg_2, \dots, Sg_n) need to be combined with roles (R_1, R_2, \dots, R_m). However, one soft gene can bind some (not always all) of the roles. Then an agent's composition can be written as:

$$\begin{aligned}
 \text{Agent} &= \langle \{\text{Soft Gene}\}, \{\text{Role}\} \rangle \\
 \text{MAS} &= \langle \{\text{Agent}\} \rangle
 \end{aligned}$$

Example 3. Code of an agent. The notes illustrate certain elements's usage. The *bind()* function maps roles to soft genes.

```

Agent a1 {
  inherit_from (a2, a3, ..., am); //get their roles, visible genes
  aggregate_from (a'2, a'3, ..., a'n); //get their roles, genes
  Soft Gene: { //divided into visible, invisible

```

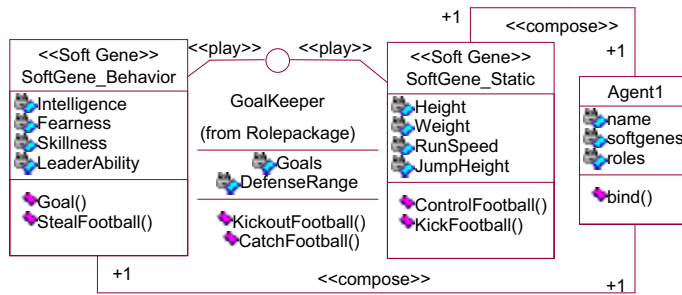


Fig. 8. The example of agent in UML

```

Visible:           //can be inherited
    SoftGene_Static g1; ...
Invisible:        //can not be inherited
    SoftGene_Behavior g2; ...
}
Role: (r1, r2, ..., rn);           //can be inherited
bind (gx, ry); }
  
```

The architecture of such agent is represented as figure 9. An agent is composed by several soft genes that bound with certain roles (e.g. role1+soft gene1, role2+soft gene2 in figure 9). 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.

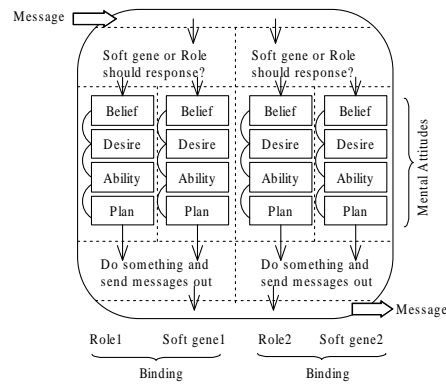


Fig. 9. The architecture of agent

3.4 Be Intelligent to Live in Open and Dynamic Systems

In an open and dynamic 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. We believe that this definition and understanding of agent has such advantages to others:

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:
 - (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 [4] or MaSE [14, 10].

4 Structure and Behavior Model of MAS

To use definitions we have given (soft gene, role, agent) to model a MAS, both structure and behavior models should be built.

4.1 Structure Model

The structure model includes:

1. Role model: The inheritance and aggregation relations among roles make up the structure model of roles. Figure 10 is an example of role model of a football team. Notice that the example is not complete for that some roles are not included in it, e.g. front, vanguard, etc.
2. Agent model: The inheritance and aggregation relations among agents make up the structure model of agents. Quite similar to role model, agent model is also a typical class diagram in UML, so we omit the graphic diagram example.

When agent a_1 inherits from agent a_2 , a_3 , etc. a_1 will inherit these agents' roles and visible genes (see example1,2,3), including all the attribute types and methods (see example1,2,3). When agent a_1 is aggregated by agent a_2 , a_3 , etc. a_1 will get all these agents' roles and genes, whether they are visible or not.

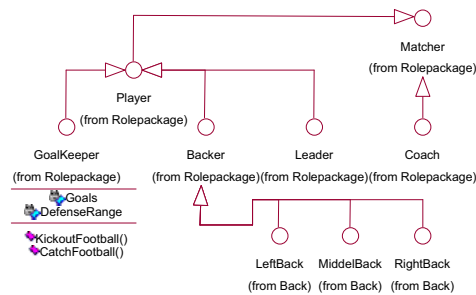


Fig. 10. The example of role model in UML

All the inheritance and aggregation relations among agents make the complete structure agent models. However, this is only simple introduction for inheritance and aggregation. In fact, we should write detailed axiomatic semantics for these two relations. The work is left for future works.

Agent structure model and role structure model together make up the system structure model.

4.2 Behavior Model

The behavior model includes:

1. Soft gene's behavior model: The methods of a soft gene stand for the gene's behaviors. Any behavior should be restricted by some behavior-rules, which are represented as pre-conditions and post-conditions in logic formulae. This model can be well represented by a typical state chart in UML. So we omit it here.
2. Agent's role transition model: An agent may bind or discard its roles at run-time. The role transition can be modelled as a finite states machine (automata), in which from the states system can know the roles bound to the agent at that time; the transitions among states happen under certain conditions. Figure 11 defines how an agent alter its bound roles, where the *note* notation is used to represent what roles an agent bind at certain state.
3. Role's interaction model: Sequence diagram may model roles' interactions well. Just a figure 12 shows, role interact model is a typical sequence diagram.

4.3 Summary

After modelling a MAS, we get such models (table 1).

However, other models should be considered in modelling pragmatic systems. For example, use case diagram [15] is one important model. We will represent a full version of models in section 5.

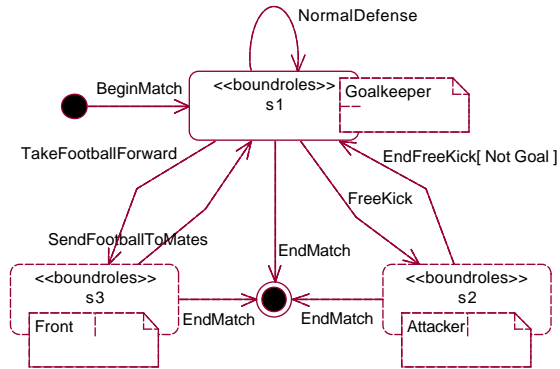


Fig. 11. The example of role transition in UML

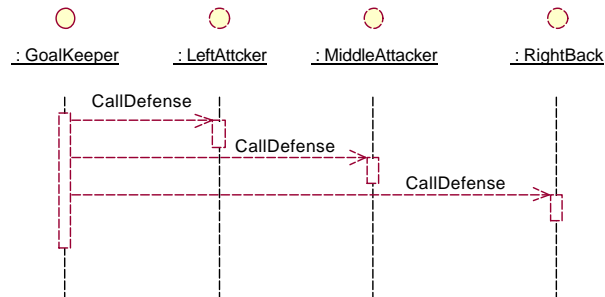


Fig. 12. The example of role interaction in UML

Table 1. Output of modelling

Structure model	Behavior model
Agent model	Soft gene behavior model
Role model	Role transition model
	Role interaction model

5 Related Works

Since MASs are often characterized as extensions of object-oriented systems, system designers always be troubled as they try to capture the unique features of MAS systems using OO tools. In response, an agent-based unified modeling language (AUML) is being developed [16, 17].

The FIPA Modeling TC's goal is to be domain independent. It has approved a work plan to develop an AUML standard.

The works of this paper belong to AUML. The general philosophy is: *When it makes sense to reuse portions of UML, then do it; when it doesn't make sense to use UML, use something else or create something new.*

6 Conclusions and Future Works

In this paper, we propose a new definition of agent based on soft gene and role. 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. The new definition leads us to an extended UML notation. The extension include:

1. New stereotypes: *play*, *compose*, *Soft Gene*, *boundroles*, whose meaning have been illustrated in sections above.
2. New diagrams:
 - (a) Agent Diagram: see figure 8. An agent is composed by some soft genes bound with certain roles.
 - (b) Role Transition Diagram: see figure 11. An agent may alter its bound roles at run-time.

Further work focuses on designing more efficient extended UML notation to support MAS modelling methodology RoMAS:

1. Soft gene and role need more explicit description on *internal attributes, external attributes, goals, services*, etc.
2. Provide semantics for modelling elements and diagrams.
3. Keep the consistency among associated models.
4. Apply the extended UML to RoboCup simulation application.

Acknowledgments

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

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. Shoham, Y.: Agent-Oriented Programming. *Artificial Intelligence* **60** (1993) 51–92
4. 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
5. Web: (www.etse.urv.es/reerca/banzai/toni/MAS/)
6. RoboCup: (<http://www.robocup.org>)
7. 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)
8. Feng, Z.Y., Sun, C.S., Wang, D.: Actor Theory: New Age and New System Calling New Human Study (in Chinese). Peking University Press (1994)
9. Iglesias, C., Garijo, M., Gonzalez, J.: A Survey of Agent-Oriented Methodologies. *Intelligent Agents V* (1999)
10. DeLoach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent Systems Engineering. *International Journal of Software engineering and Knowledge Engineering* **11**(3) (2001)
11. Staff, A.H.: *The Amercian Heritage Dictionary*. Turtleback Books (01/01/2001)
12. Web: (<http://www.accessexcellence.org/AB/GG/genetic.html>)
13. Yan, Q., Mao, X.J., Qi, Z.C.: Modeling Role-Based Organization of Agent System. *UKMAS'02* (2002)
14. DeLoach, S., Wood, M.: Developing multiagent systems with agenttool. *Intelligent Agents VI* **1757** (2000)
15. Jacobson, I.: *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison Wesley (1992)
16. : Auml web site, www.auml.org. Technical report (2003)
17. Koning, J.L., Romero-Hernandez, I.: Generating machine processable representations of textual representations of auml. In: *AAMAS Workshop on Agent-Oriented Software Engineering (AOSE)*, Bologna, Italy (2002)