

ROLE-BASED MULTI-AGENT SYSTEMS

Haibin Zhu

Department of Computer Science and Mathematics, Nipissing University
100 College Drive, North Bay, Ontario, P1B 8L7, Canada
Tel: +1(705)474-3450 ext. 4434
Fax: +1(705)474-1947
E-mail: haibinz@nipissingu.ca

MengChu Zhou

Department of Electrical and Computer Engineering, New Jersey Institute of Technology,
University Heights, Newark, NJ 07102, USA
Tel: +1(973)596-6286
Fax: +1(973) 596-5680
E-mail: zhou@njit.edu

ABSTRACT

Agent system design is a complex task challenging designers to simulate intelligent collaborative behavior. Roles can reduce the complexity of agent system design by categorizing the roles played by agents. The role concepts can be also used in agent systems to describe the collaboration among cooperative agents.

This chapter introduces roles as a means to support interaction and collaboration among agents in multi-agent systems. It reviews the application of roles in current agent systems at first; then it describes the fundamental principles of role-based collaboration and proposes the basic methodologies of how to apply roles into agent systems, i.e., the revised E-CARGO model; after that, it demonstrates a case study: a soccer robot team designed with role specifications; and at last, this chapter presents the potentiality to apply roles into information personalization.

KEYWORDS

Role, Role-Based, Agent, Multi-Agent Systems

ROLE-BASED MULTI-AGENT SYSTEMS

ABSTRACT

Agent system design is a complex task challenging designers to simulate intelligent collaborative behavior. Roles can reduce the complexity of agent system design by categorizing the roles played by agents. The role concepts can be also used in agent systems to describe the collaboration among cooperative agents.

This chapter introduces roles as a means to support interaction and collaboration among agents in multi-agent systems. It reviews the application of roles in current agent systems at first; then it describes the fundamental principles of role-based collaboration and proposes the basic methodologies of how to apply roles into agent systems, i.e., a role-based model; after that, it demonstrates a case study: a soccer robot team designed with role specifications; and at last, this chapter presents the potentiality to apply roles into information personalization.

KEYWORDS

Role, Role-Based, Agent, Multi-Agent Systems

INTRODUCTION

Artificial Intelligence is the discipline aimed at understanding intelligent beings by constructing intelligent systems (Castelfranchi, 1998). From a behaviorist's perspective, intelligent systems are those that can simulate human beings' work that requires intelligence including logic reasoning, problem solving, deduction and induction. A distributed system is composed of many computers inter-connected via communication networks, which cooperate and coordinate to accomplish a common task or goal (Coulouris *et al.*, 2005). Multi-agent systems are intelligent systems built on a distributed computer system. They are based on the use of cooperative agents and organized with hardware/software components. In such systems, each agent independently handles a small set of specialized tasks and cooperates to achieve the system-level goals and a high degree of flexibility (Ahn, 2003; Gruver, 2004). Multiagent systems are becoming more relevant to artificial intelligence (AI) (Bowling, 2002) and can be used to implement distributed AI (DAI). The agent concept evolves from objects. It is a combination of object-orientation and AI. The AI community claims intelligence as a natural quality of agents and uses the traditional symbolic representation to describe agents. Maes (1994) defines agents as computational systems that inhabit a complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed. Wooldridge (1995) and Jennings (1996) define agents as hardware-based or software-based computer systems that possess the following properties:

- autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- social ability: agents interact with other agents (and possibly humans) via a kind of agent-communication language;
- reactivity: agents perceive their environment, which may be the physical world, a user via a graphical user interface, a collection of other agents, or perhaps all of these combined, and respond in a timely fashion to changes that occur in it;
- pro-activeness: agents do not simply act in response to their environment. They are able to exhibit goal-directed behavior by taking an initiative.

Normal objects can be thought of as passive because they wait for a message before performing an operation. Once invoked, they execute their method and go back to "sleep" until the next message arrives. A current trend is to design objects that not only react to

events in their environment, but also behave proactively. Therefore, in addition to traditional object properties, an agent should also have the following characteristics:

- Active, i.e., an agent may act according to its internal states and goals. Note that an object in its conventional meaning can only respond to the messages sent to it even though many acclaim that everything is an object;
- Autonomous, i.e., an agent is not controlled directly by people; and
- Collaborative, i.e., agents need to collaborate with others to accomplish a complex task.

Every agent is responsible for accomplishing a certain task. It can be considered as a self-contained object of some class and involves itself with a specific environment. The environment exists before agents are created. Agents should be designed such that they can adapt to a constantly changing environment.

Agent design is a complex task challenging designers to simulate intelligent human behavior. Searching and retrieving are considered expressions of intelligence. Roles can reduce the complexity of agent design by categorizing agent responsibilities. Based on the theory of search and retrieval, such separation greatly shrinks the search space and tends to significantly increase agent efficiency in response to messages or events relevant to a specific role.

In fact, multi-agent systems are simulations of human societies or virtual societies. To simulate real societies, we need to understand their nature. A well-organized society should encourage member contributions (Mills and Simmons, 1999). Similarly, distributed intelligent systems should establish a healthy platform or environment for virtual participants who contribute and work effectively. One can consider agents as virtual citizens in intelligent systems. A capable political system promotes a harmonious and flourishing society. Good architecture leads to a beautiful and long-lasting building. Clearly, a distributed intelligent system requires good system architecture. A role-based design can provide such architecture. Organizational and social psychological theory, suggests that an organization is composed of three key elements: participants, goals and roles (Cyert, 1968). A goal is a mental representation of a process, which expresses an agent's internal anticipation and regulations (Castelfranchi, 1998). Therefore, it is meaningful to use roles to express a participant's goal in an organization (Zhu, 2005). Furthermore, it can be said that roles, role relationships, role players (agents) and role playing constitute all the aspects for distributed intelligent systems. Roles are commonly applied concepts in many fields, such as sociology, psychology, social psychology, behavioral science, management, and drama. Roles can be applied into people's natural organizations, task distributions, and system analysis, system design and system construction (Alon, 2003). Therefore, the introduction of roles has many potential benefits such as encouraging people's contributions to collaboration (Zhu, 2005); even task distribution; evolutionary expression; reuse, and the reduction of agent complexity.

An important goal of multi-agent systems is to obtain collective intelligence (Heylighen, 1999) through the organization of numerous agents. This chapter discusses the fundamentals of Role-based Multi-Agent Systems (RMAS) by applying roles into the design and implementation of multi-agent systems. This work is based on our previous one, i.e., the Environments, Classes, Agents, Roles, Groups and Objects (E-CARGO) model (Zhu and Zhou, 2006a).

This chapter is arranged as follows. Section 2 reviews the applications of role concepts in agent systems; Section 3 describes the revised E-CARGO model for RMAS; Section 4 demonstrates a role-based architecture for multi-agent systems; Section 5 discusses the agent evolution expressed by roles; Section 6 discusses the benefits and challenges of RMAS; and Section 7 concludes the chapter and proposes topics for future research.

REVIEW OF ROLE CONCEPTS IN AGENT SYSTEMS

Betcht *et al.* (1999) propose an agent system ROPE which includes roles. They point out that roles provide a well-defined interface between agents and cooperative processes. This allows an agent to read and follow, normative rules established by the cooperation process even if not previously known by the agent. Their major motivation to introduce such roles is to increase the agent system's adaptability to structural changes. They formally define a role as an entity consisting of a set of required permissions, a set of granted permissions, a directed graph of service invocations, and a state visible to the runtime environment but not to other agents. Roles in ROPE are used to describe the separate behavior of an agent. Their paper describes the architecture of ROPE but lacks a prototype system to demonstrate the practicability and usability of this architecture.

Stone and Veloso (1999) point out that a role consists of a specification of an agent's internal and external behavior. They state that roles may be rigid, i.e., completely specifying an agent's behavior or flexible, i.e., leaving a certain degree of autonomy to the agent filling the role. They introduce role concepts from their research on AI and model them via a robot soccer team.

Depke *et al.* (2001) study role concepts in agent systems. In their approach, roles are modeled as traditional classes. Thus roles are the latter's instances. Roles encapsulate certain tasks, responsibilities and goals of an agent. They conclude that roles can be applied into agent systems for the following purposes: 1) to express the organizational structure of a multi-agent system; 2) to specify interactions in a generic way; and 3) to serve as agent-building blocks in class diagrams. A short-coming of this approach is that a role is deleted when the adopting agent is destroyed. This can be an inaccurate depiction of reality. For example, should the President of a country die while in office, people must elect another but not eliminate that societal role.

In agent-oriented software engineering, Gaia methodology (Wooldridge *et al.*, 2000; Zambonelli *et al.*, 2003) is proposed to support system analysis and design by taking a multi-agent system as an organization. Analysis and design are well-separated phases, i.e., analysis aims to develop an understanding of the system structures through role and interaction models while the design phase aims to define agent details within the system. An important contribution of Gaia is in modeling roles that accommodate agent rights or permissions. In Gaia, roles are described with responsibilities, permissions, interaction protocols and activities. The ideas on role application in agent-oriented software engineering are similar to those expressed in our previous work (Zhu 2006d; Zhu and Zhou, 2006b).

In Organization-Centered Multi-Agent Systems (OCMAS) (Ferber *et al.*, 2004), roles are emphasized as an important element of organizations. A role is the abstract representation of a functional agent position in a group. It helps overcome the drawbacks of agent-centered multi-agent systems, i.e., an agent is open for the entire system; there are no constraints on inter-agent accessibility; and agent interaction occurs directly. The roles in OCMAS are similar to those in (Wooldridge *et al.*, 2003). Group roles, as mentioned in (Ferber *et al.*, 2004), refer to roles in a group, i.e., an agent must play a role in a group. This definitely incurs an argument of which should be based on the other: roles or groups (Zhu and Zhou, 2006a).

Cabri *et al.* (2005a; 2005b) discuss how to introduce roles into agent systems. The role ideas in both Role-Based Access Control (RBAC) and modeling methodologies are reviewed. Supporting roles at the implementation level is emphasized. Roles can be exploited by agents at runtime in order to enhance their capabilities. A role can be thought of as a set of behaviors and capabilities that agents can exploit to perform their tasks in a given context. A role is

temporary in that an agent must perform it within a specified period of time or context. It is generic, in the sense that it is not tightly bound to a specific application, but expresses general properties that can be used in different applications by different agents. Roles are related to contexts so that each environment can impose its own rules and can grant some local capabilities, forcing agents to assume specific roles. A role is described by four elements in the XRole language (Cabri *et al.*, 2002; 2005), i.e., name, description, keyword and action. A unique name precisely identifies a role. The description allows designers to understand role objectives in human-readable sentences. Keywords can also be used in role identification. Actions allow for interactions among agents or execution environments. Roles in agent systems are significant on two important fronts. From an environmental point of view, a role imposes a defined behavior on the entities assuming it. From the application perspective, it expresses a set of capabilities, which can be exploited by agents in carrying out their tasks; and it can be used to support the separation of concerns of agents (Botha and Eloff, 2001).

It is arguable that roles are temporary. We can say that a person may play a role temporarily. However, the lifetime of a role is often longer than an object or agent. For example, Mr. Bill Clinton played a role as USA President for eight years while there is always the role called USA President. A role serves as an abstract description of functions which must be fulfilled in reaching an assigned goal. In the agent-oriented approaches, roles are a proper means of refining agent-oriented modeling. However, they still lack a clear definition, e.g., how a role is defined for a special agent. They still consider a role as a property of an agent that is similar to the ideas in object modeling. There are some important aspects of roles left unconsidered.

Odell *et al.* (2003; 2005) consider the aspects of roles from the viewpoint of theaters and behavioral science. They refer to Biddle and Thomas (1966)'s and other psychologists' work and cite Shakespeare's role concepts in theaters. They state that roles specify normative behavioral repertoires for agents and provide both the building blocks for social agents and the requirements by which agents interact. They emphasize the diversity and limited capability of the roles agents can play.

Partsakoulakis and Vouros (2004) survey role concepts and mechanisms in multi-agent systems. Roles are viewed as tools to manage the complexity of tasks and environments. Roles are intuitively used to analyze agent systems, model social activities and construct coherent and robust teams of agents. High degree of interactions, environment changes and distributivity of roles are emphasized.

Roles are a useful concept in assisting designers and developers with the need for interactions. They can help system designers focus on those conditions where social determinants are more influential (Odell, 2003). Roles help the agent-application developers/designers to model the execution environment. They also allow agents to actively recognize the environment themselves (Cabri, 2004).

Agent systems are currently among the most active fields in role application. Most approaches to agent-oriented modeling use only a few of the possible aspects of roles. Role concepts are generally used in agent systems to describe the collaboration among cooperative processes or agents. We call these roles agent-roles.

In agent systems, the following principles related to a role are widely accepted:

- 1) A role instance is deleted when an agent is destroyed, i.e., its lifetime depends on its agents.
- 2) Roles are used to form different interfaces for agents in order to restrict the visibility of features and to handle permissions for the access to the internal state and role services of agents.

- 3) Roles have three functions: comprise special behavior, form the behavior of an agent, and take a position in a group of agents.
- 4) A role specifies a position and a set of responsibilities that are made up of services and tasks.
- 5) Roles are modeled as stereotyped classes.
- 6) Roles can be used for expressing the organizational structure of a multi-agent system.
- 7) Roles can be used for specifying interactions in a generic way.
- 8) Roles can be used as agent-building blocks in class diagrams.

From Table 1, one can obtain a good picture of the literature on roles as modeling mechanisms in agent systems.

Table 1: Role as Modeling Mechanisms in Agent Systems

Year	Authors	Motivations	Contributions	Conclusions
1999	Becht	To increase the agent system's adaptability to structural change	Propose an architecture ROPE to support roles in agent systems	Roles can be used to support the cooperative processes among agents.
1999	Stone Veloso	To provide a well-defined team structure	Propose a teamwork structure with roles	Agents can respond to changing environments by dynamically changing their roles.
2001	Depke Heckel Kuster	To analyse the application of roles in agent systems	Analyze the role requirement and propose a method to transform UML diagrams into code	Roles can be used in many ways in modeling agent systems, such as expressing an organizational structure, specifying interactions, and being agent-building blocks.
2003	Odell, <i>et al.</i>	To simplify the design of sophisticated agent systems	Enforce that the roles for agent systems should be consistent with those in behavioral science	Roles are good implications of how agents behave in a group.
2004	Partsakoulakis Vouros	To support the collaboration among agents	Emphasize the importance of roles for agent systems by analyzing the properties of roles in agent systems.	The extensive use of roles in implemented systems shows the need for role-oriented thinking and modeling in agent system development.
2004	Cabri Ferrari Leonardi	To investigate how the concept of role can be exploited in agent systems and how to simplify the related tasks	Propose the framework BRAIN that exploits the concept of role in different phases of the development based on a simple yet general role-based model for interactions	The use of roles can bring different advantages, in terms of separation of concerns between algorithmic issues and interaction issues, generality of approaches, locality, and reuse of solutions and experiences.
2004	Ferber	To overcome the drawbacks of agent-centered multi-agent systems	Propose organization-centered multi-agent systems by introducing roles	The OCMAS with roles overcomes the drawbacks of agent-centered multi-agent systems.
2005	Cabri Ferrari Leonardi	To support agent interactions and increase agents' adaptabilities	Implement a role interaction infrastructure that enables Java agents to dynamically assume and use roles at runtime	Roles can be more useful to design, develop, and even maintain complex applications, where there are many interactions among interacting agents.
2005	Odell Nodine Levy	To design a metamodel for agents and groups	Continue their work of 2003 and propose a metamodel for agents, roles and groups	The metamodel enhances the predictability, reliability and stability of agent systems.

FUNDAMENTAL PRINCIPLES OF RMAS

Roles are appropriate tools for designing multi-agent systems since they can provide platforms for agents to execute their tasks. A role describes both the service requirement and the ability to provide services. With provided services, an agent does not need to bring with it many things that are required by traditional mobile agents. In a traditional method, an agent should be able to have their own facility to provide required services. Role-based methods reduce further the traffic requirement for mobile agents.

Although role concepts and agent systems have been investigated for decades, their combination attracted attention only several years ago. Only recently, in an agent community, have roles been considered as an important concept and mechanism in assisting agents' interaction, coordination and collaboration.

Although there is a common belief that roles are important concepts, until now no consensus has been reached as to how roles should be represented and integrated into agent systems. Even though role theory has been investigated in social psychology for several decades, there is neither a unified theory nor commonly accepted specification tool for roles. Roles can be taken as a highly abstract concept that is very flexible for players to enact. For example, a *manager* role is highly abstract, different people taking the role *manager* differently. On the other hand, roles can be a very concrete process description for players to follow. For example, a labor worker role in a manufacturing production line is to perform rigid operations and processes. Its players must conduct these operations and processes. There is little or no flexibility in playing such a role.

From the literature relevant to roles, the following aspects of roles are discussed (Fig. 1):

Considering role expression, we have:

- **Rights:** Roles are entities that facilitate users (agents) in accessing system resources (files, objects, and devices). Roles are tags or tickets attached to users to access objects.
- **Responsibilities:** Roles are entities that express different aspects of an object at different context and time. They provide different services to the outside world. The rights of an object are taken granted of all the objects in its context or scope.
- **Both rights and responsibilities:** in social psychology, people who live in a society should take responsibilities and hold rights when playing a role.

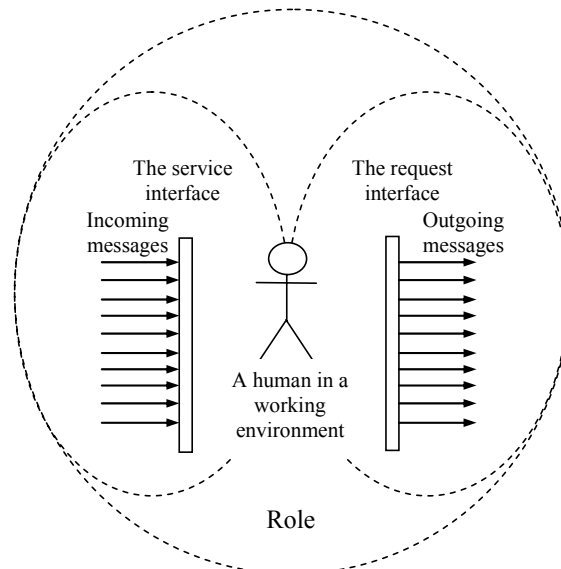


Fig. 1 The different views of roles

On the subject of concrete role specification, we have:

- Interfaces: Roles are abstract entities to express the interfaces between objects, agents or people in collaboration. In this sense, roles only specify what the services are and what the requests are. How the services and requests are processed depends on their players. We can call them as interface-roles.
- Processes: Roles are concrete behavior describers in specifying the functions of an object, agent or human. They specify not only what services and requests are but also how services and requests are processed. We can call them process-roles.

To support role-based agent collaboration, it is necessary to construct a role-based agent system that allows for agent cooperation and successful collaborative results. We must understand the fundamental principles with which we need to comply. They can be established based on many principles learned from modeling methodologies, software engineering and intelligent system development.

With object-oriented principles, we can conceptually construct the underlying components of a role-based system including classes, objects, messages, interfaces, agents, roles, and groups. We can use agents to simulate human behaviors. The following four sets of principles related to object, agent, role and group (Zhu, 2006a; Zhu and Zhou, 2006a) must be addressed when one builds RMAS.

Object principles

Object-oriented methodologies are widely used in system modeling and software engineering. A multi-agent system can also be considered as an object system at first. Its development should obey the following object principles (Meyer, 1988; Kay, 1993; Zhu and Zhou, 2003):

- O1: Everything in the world is an object. An object can be used to express everything in a collaborative system.
- O2: Every system is composed of objects and a system is also an object.
- O3: The evolution and development of a system is caused by the interactions among the objects inside or outside the system.
- O4: Objects can be created, modified, and deleted.
- O5: A message is a way to activate services of an object.
- O6: An interface is a list of message patterns.
- O7: The interactions among objects are expressed by sending messages that are requests to invoke objects' actions.
- O8: Each object is an instance of a class. A class shows the commonality of a group of objects.
- O9: A class X may inherit another class Y. Y is called a superclass while X is called a subclass.
- O10: Classes can be taken as templates of objects.

Agent principles

To make real multi-agent systems and overpass object systems, agents should be distinguished from objects. The agent principles are as follows:

- A1: Agents are special objects that simulate the behavior of people.
- A2: Agents can be created, modified, and deleted.
- A3: Agents are autonomous. They should be able to reply incoming messages and send outgoing messages based on their situations (Etzioni and Weld, 1995; Jennings *et al.*, 1998; Russell and Norvig, 2003).
- A4: Agents are adaptive. They should be able to understand their environment and take actions to change the environment and make it better for them to live (Etzioni and Weld, 1995; Russell and Norvig, 2003).
- A5: Agents are social. They should be able to interact with other agents (Jennings *et al.*, 1998; Russell and Norvig, 2003).

- A6: Agents are collaborative. They may join a group to work for a common goal or quit a group if they do not want to cooperate more.
- A7: Agents are flexible. Not all actions of agents are predicted. They can dynamically choose which actions to invoke, and in what sequence, in response to the state of its environment (Jennings *et al.*, 1998; Russell and Norvig, 2003).
- A8: Agents are mobile. They are able to transport them from one site to another in a system (Etzioni and Weld, 1995).

Role principles

Based on the discussion of Section 2, we derive the following role principles:

- R1: A role is independent of agents.
A role should be defined separately. It is commonly understood that a role depends on objects in object systems. In collaboration, however, collaborators do not necessarily care about who services them as long as the service is performed to their expectation. For example, professor X in a university may ask for services from the technology service department. X does not care who provides the service so long as they are performing the role of technician.
- R2: A role can be created, changed and deleted.
A role includes both responsibilities (the service interface) when an agent is taken as a server and rights (the request interface) when the agent is taken as a client. To specify a role means to specify both responsibilities and rights. A role does not accomplish the tasks specified by the responsibilities. Only agents playing the role accomplish the tasks.
- R3: Roles can be interface-roles.
As for the service interface, a role is actually a filter of messages sent to an agent. This filter only allows passage of the messages relating to the role played by the agent. As for the request interface, a role expresses or restricts the accessibility of an agent to the system.
- R4: Roles can be process-roles. In such roles, what to do, how to do it, and what to access are all rigidly specified.
- R5: Roles are taken as media for interactions.
Interactions among agents are based on their roles, i.e., a message to request collaboration with other agents is sent to the relevant roles they are playing. Roles emphasize that the message receivers are firstly roles but not agents. Any available agent playing a role can respond to the messages received by the role.
- R6: Playing a role means that the agent is attached to a role. A role can be played by one or more agents at the same time.
- R7: An agent may adopt one or more roles but can perform only one at a time.
- R8: Roles can be used to support indirect and direct interactions.
For direct interactions, each role has exactly one agent to play. For indirect interactions, each role has multiple agents to play. In the former case, identifying a role means identifying an agent. In the latter, identifying a role does not mean identifying an agent.
- R9: Roles can have hierarchy relationships. Higher level roles can be taken as goals for agents playing roles at lower levels.
In long-term collaboration, autonomous agents such as human beings may hope to play higher level roles as their careers advance.

Group principles

In reality, people work in a group and may hold multiple roles. Every work setting involves groups of individuals. To accomplish a common task, the group members, i.e., agents, interact with each other. We should follow the principles as follows.

G1: A group is necessary to build a multi-agent system.

G2: A group can be created, changed and deleted (Chockler *et al.*, 2001).

G3: Before specifying a group, we must specify all the roles in it.

G4: Group formation allows agents to join the group and play roles. They are named as the members of this group.

G5: A group can be embedded, i.e., one group may be an object in another (Tanenbaum and van Steen, 2002).

G6: A group can be overlapped with other groups, i.e., the members may belong to two or more groups (Tanenbaum and van Steen, 2002).

G7: A group can be public or private (WebBoardTM, 2006).

Public means that all the agents in the system can join the group. Private means that joining a group is controlled by a special agent who plays a special role called moderator.

G8: A group can be open or closed (Coulouris *et al.*, 2005).

Open groups mean that new agents are permitted to join the groups while closed ones allow no further membership.

The above four sets of principles are reflected in the revised E-CARGO model to be discussed next.

THE REVISED E-CARGO MODEL FOR RMAS

To collaborate, people generally join a group or organization. All individuals should have clear positions within a group and their roles should be related without causing interference (Zhu and Zhou, 2006a). Unclear role specification may create dysfunctional ambiguity and conflict in an organization (Bostrom, 1980), while clear role definition and specification can help people collaborate effectively in a group (Becht *et al.*, 1999). Role conflict and ambiguity are causes of stress in organizations. Role conflict means situations in which individuals do not know how they should behave due to differing expectations. Role ambiguity means a situation in which individuals do not know exactly how they are expected to behave based on of vague, abstract expectations (Ashforth and Mahwah, 2001; Miner, 1992). A collaborative learning environment may have to establish and assign well-defined roles to participants to foster interaction (Singley, 2000). Dynamic role assignment requires a stable basis to improve productivity and performance. By “dynamic” we mean that role assignment and reassignment occur during collaboration. People may be expected to dynamically change their roles according to the needs of the group and organization (Dafoulas and Macaulay, 2001; Ould, 1995). Therefore, roles are required to be taken as underlying mechanisms in collaboration, yielding a concept of role-based collaboration (Zhu, 2006a; Zhu and Zhou, 2006a; Zhu and Zhou, 2006b).

E-CARGO (Zhu and Zhou, 2006a) model is proposed to support collaboration among people by using computer systems. It formulates role-based collaboration by providing underlying components such as classes, objects, agents, roles, environments and groups. With the support of E-CARGO model, people can practice role-based collaboration easily and naturally.

Based on the principles discussed in Section 3, the E-CARGO model needs to be revised to support RMAS. In this revised model, human users \mathcal{H} in the original E-CARGO model are removed to emphasize the autonomy of agents.

Object and Class

From the viewpoint that everything in the world is an *object* and every object has a *class*, an object must be (Kay, 1993; Meyer, 1988; Zhu and Zhou, 2003)

- “uniquely identified. i.e., any object should carry a unique identification;
- “created or destroyed”; and
- “communicative. i.e., an object can exchange messages with other objects.

It may be

- “nested. i.e., a complex object has other objects as its components (which in turn may have object components);
- “active and autonomous. i.e., an object may respond to messages without people’s intervention; and
- “collaborative. i.e., collaborative relationships among objects arise when they exchange messages.

We use agents to express the objects that possess all the properties discussed above.

Considering the general meaning and properties of objects, we can express a *class* by a quadruple.

Definition 1: class. $c ::= \langle n, \mathcal{D}, \mathcal{F}, \mathcal{X} \rangle$ where

- n is the identification of the class;
- \mathcal{D} is a data structure description for storing the state of an object;
- \mathcal{F} is a set of the function definitions or implementations; and
- \mathcal{X} is a unified interface of all the objects of this class. It is a set of all the message patterns relevant to the functions of this class. A message pattern tells how to send a message to invoke a function.

We use c to express a specific class and \mathcal{C} the set of all classes. Next, we define an *object* based on a class.

Definition 2: object. $o ::= \langle n, c, s \rangle$ where

- n is the identification of the object;
- c is the object’s class identified by the class identification or name; and
- s is a data structure whose values are called attributes, properties, or states.

We use o to express a specific object and \mathcal{O} the set of all the objects. The above two definitions comply with object principles O1-9.

Agent

An agent is defined as an entity consisting of a set of provided services (Castelfranchi, 1998).

An *agent* a is a special object that can simulate the intelligent behavior of human being. It is different from objects in that it responds to messages based on its current state but conventional objects respond to messages directly.

Definition 3: agent. $a ::= \langle n, c_a, s, r_c, \mathcal{R}_p, \mathcal{N}_g, e_t, e_s, \varphi, u \rangle$, where

- n and s have the same meanings as those in Definition 2;
- c_a is a special class that describes the common properties of agents;
- r_c means a role that the agent is currently playing. If it is empty, then this agent is free;
- \mathcal{R}_p means a set of roles that the agent is potential to play ($r_c \notin a. \mathcal{R}_p$);
- \mathcal{N}_g means a set of identifications of groups that the agent belongs to;
- $\langle e_t, e_s \rangle$ expresses the processing capacity for an agent, where e_t expresses how many units of free time it has and e_s expresses how much memory space it has. $\langle e_t, e_s \rangle$ can be reset based on the performance of an agent’s services. Even though the time in one

day is 24 hours, a person may have different e_t and e_s based on their processing capacities including the working efficiency, attitudes, and goals;

- φ expresses the past performance or credits of serving others; and
- u to expresses the workload of the agent.

This definition complies with agent principles A1-8. c_a and s are used to reflect A1-2. $\mathcal{N}_r, \mathcal{N}_g, e_t, e_s, \varphi$, and u are used to reflect A3-8. To support its functionality, an agent should have the knowledge about groups, classes, objects and other agents in the system. \mathcal{A} denotes the set of all agents.

Message

Interaction is a necessary entity for collaboration. To facilitate interactions among roles while following the principles O3-5, messages are used.

Definition 4: message. $m ::= \langle n, v, l, \mathcal{P}, t, \varphi \rangle$ where

- n is the identification of the message;
- v is null or the receiver of the message expressed by an identification of a role;
- l is the pattern of a message, specifying the types, sequence and number of parameters;
- \mathcal{P} is a set of objects taken as parameters with the message pattern l , where $\mathcal{P} \subset \mathcal{O}$;
- t is a tag that expresses any, some or all-message and
- φ is the weight for an agent to collect its credit for promotion. If an agent responds to this message, the agent receives the weight and adds it to its credit.

In a traditional object model, we concentrate mainly on objects and their classes because executable programs run automatically with little interaction with users at the system level. A traditional object-oriented paradigm emphasizes the messages accepted by a class of objects. However, it does not give much consideration to the messages an object may send out. In this model, we emphasize that a role is a message receiver and a user sends messages through roles. In this definition, if v is null, this message is an incoming one to be dispatched by the role to an agent and \mathcal{P} and t are meaningless. If v is an identification of a role, it is an outgoing message that should be dispatched by the role specified by the message. Outgoing messages should be filled with \mathcal{P} (it can be empty) and t . We divide the outgoing messages into three categories by t , *any-message*, *some-message* and *all-message*. By any-message we mean that the message may be sent to any agent who plays the role. Some-message means the messages should be sent to some agents playing the role and all-message means that the messages should be sent to all the agents who play the role.

We use m to express a message and \mathcal{M} the set of all messages. We call m a message template if its \mathcal{P} is not specified completely.

Role

A *role* can show the specialties of some users. It provides them with not only message patterns to serve others but also message patterns to access objects, classes, groups and other roles.

Definition 5: role. $r ::= \langle n, I, \mathcal{N}_a, \mathcal{N}_o, e_t, e_s, \mathcal{R}_m, \mathcal{R}_g, \varphi \rangle$ where,

- n is the identification of the role;
- $I ::= \langle \mathcal{M}_{in}, \mathcal{M}_{out} \rangle$ denotes a set of messages, wherein, \mathcal{M}_{in} expresses the incoming messages to the relevant agents, and \mathcal{M}_{out} expresses a set of outgoing messages or message templates to roles, i.e., $\mathcal{M}_{in}, \mathcal{M}_{out} \subset \mathcal{M}$;
- \mathcal{N}_a is a set of identifications of agents that are playing this role;

- \mathcal{N}_o is a set of identifications of objects including classes, environments, roles, and groups that can be accessed by the agents playing this role;
- e_t and e_s are used to express the processing capacity requirement, where e_t expresses how many units of free time it requires and e_s expresses how many units of space it requires. $\langle e_t, e_s \rangle$ expresses that an agent must possess at least $\langle e_t, e_s \rangle$ to play this role;
- \mathcal{R}_m is the super roles;
- \mathcal{R}_s is the subordinate roles; and
- φ is the required credits for an agent to play this role.

This definition follows role principles R1-10. I is used to reflect R3-7. \mathcal{N}_a and \mathcal{N}_o are used to reflect R8-10.

In a run-time system, \mathcal{M}_{in} is a subset of \mathcal{X} (interface) of C_a (set of special classes describing common properties of users) of an agent a that plays this role. The elements of \mathcal{M}_{out} are constructed with the sub-sets of \mathcal{M}_{in} of other roles. Suppose that we have at least one agent playing a role, for roles r_i and r_j ($i \neq j$),

$$\forall m_{in} (m_{in} \in r_i.M_{in} \rightarrow \exists a (m_{in} \in a.C_a.X \wedge a.n \in r_i.N_a))$$

$$\forall m_{out} (m_{out} \in r_i.M_{out} \rightarrow \exists r_j (i \neq j, \exists m_{in} (m_{in} \in r_j.M_{in} \wedge m_{in}.I = m_{out}.I)))$$

where m_{in} expresses an incoming message $m_{in} \in M_{in}$ and m_{out} an outgoing message ($m_{out} \in M_{out}$).

In Definition 5, accessing means obtaining all the services the objects provide. Note that we separate agents from objects to emphasize that a role is the media to access agents. The only way to interact with other agents is by sending messages to roles. Denote by \mathcal{R} the set of all roles.

\mathcal{N}_a in Definition 5 and \mathcal{N}_r in Definition 3 are just sets of identifications of agents and roles.

They might be empty at the beginning. They are used to express the dynamic properties of roles and agents. If $|N_r|=0$, the user represented by the agent is playing no role. If $|N_a|=0$, the role has not been played by any agent. When we issue a message $r.addAgent(a)$, we mean that we add the identification of agent a to role r and the identification of r to a .

To facilitate role playing, roles should constitute a hierarchy. A role hierarchy is actually a partial order on roles \mathcal{R} and a relation $>$, i.e., $(\mathcal{R}, >)$. $r_1 > r_2$ means that r_1 is a super role of r_2 . To express this hierarchy, we need an item in the role entity to accommodate the identifications of subordinate roles and super roles. Note that this relationship expresses the promotion direction for roles.

Environment

In reality, people collaborate in an environment. People normally build groups in an environment. We can mimic a play on a stage. The stage is the environment. The play or collaboration is performed by a group of actors. Therefore, we introduce a new concept to facilitate the definition of a group.

Definition 6: environment. $e ::= \langle n, \mathcal{B} \rangle$ where

- n is the identification of the environment; and
- \mathcal{B} is a set of tuples of role, number range and an object set, $\mathcal{B} = \{ \langle n_r, q, \mathcal{N}_o \rangle \}$. The number range q tells how many users may play this role in this environment and q is expressed by (lower, upper). For example, q might be (1, 1), (2, 2), (1, 10), and (3, 50). It states that how many agents may play the same role r in the group. The object set \mathcal{N}_o consists of the objects accessed by the agents who play the relevant role. By “complex” we mean they are composed of other objects. The complex objects in \mathcal{N}_o are mutually exclusive, i.e., one complex object in this set can only be accessed by one agent (user). For each tuple, we have the inequality: $q.lower \leq |N_o| \leq q.upper$. In fact, $|N_o|$ expresses the number of resources for agents to access.

For example, a computer science department can be expressed as an environment:

- n = Department of Computer Science;
- \mathcal{B} has five tuples as its members, i.e., $\mathcal{B} = \{ \langle \text{chairperson}, [1, 1], \{ \text{a chairperson office} \} \rangle, \langle \text{associate chairperson}, [1, 3], \{ \text{one to three offices for associate chairpersons} \} \rangle, \langle \text{secretary}, [1, 5], \{ \text{one to five offices for secretaries} \} \rangle, \langle \text{computer administrator}, [1, 4], \{ \text{one to four offices for administrators} \} \rangle, \langle \text{faculty}, [3, 15], \{ \text{three to fifteen faculty offices} \} \rangle$. It implies that one chairperson, one chairperson office; one to three associate chairpersons and one to three offices for associate chairpersons; one to five secretaries and one to five offices for secretaries; one to four computer system administrators and one to four offices for administrators; three to fifteen faculties and three to fifteen faculty offices.

Denote by \mathcal{E} the set of all the environments in a system. All the roles in an environment have access to it. With this definition, we know that prior to environment creation, roles must be specified. At the same time, before we specify a role, we need to create objects for the roles to access.

Group

Agents work in a group and hold roles. Every work setting involves groups of individuals. In a group, to accomplish a task, the group members (agents) interact with each other. We can define a group as a set of agents playing roles in an environment.

Definition 7: $g = \langle n, e, J \rangle$ where

- n is the identification of the group;
- e is an environment for the group to work; and
- J is a set of tuples of identifications of an agent and role, i.e., $J = \{ \langle n_a, n_r, n_o \rangle \mid \exists q, n_o (n_o \in N_o) \wedge (\langle n_r, q, N_o \rangle \in e.B) \}$.

Definitions 6-7 comply with group principles G1-8. n and e are used to reflect G1-3; e and J are used to reflect principles G4 and G5.

Suppose U ($\{ \{ n_a \mid \exists n_r, n_o (\langle n_a, n_r, n_o \rangle \in J) \} \}$) agents in the group, V ($\{ \{ n_r \mid \exists n_a, n_o (\langle n_a, n_r, n_o \rangle \in J) \} \}$) roles in the environment e of a group and one agent plays exactly one role in the group. q_i means the role number range for the i th role, $q_i.upper$ means the upper limit of the number of the agents playing the i th role and $q_i.lower$ means the lower limit, we have the following inequality for a group:

$$\sum_{i=1}^V q_i.lower \leq U \leq \sum_{i=1}^V q_i.upper$$

It means that every agent must play a role in a group, an agent may join the group only when vacancies are available for a role, and there must be a number of agents to play relevant roles.

It also follows principle G8, i.e., an open group. The group is closed if $U = \sum_{i=1}^V q_i.upper$.

If we want the resources in an environment to be fully used without waste, we should keep the equality, $\sum_{i=1}^V b_i \cdot |N_o| = |J|$ for each $b_i \in B$ in group g .

Clearly, $\sum_{i=1}^V b_i \cdot |N_o| > |J|$ means that there are more resources than the required while

$\sum_{i=1}^V b_i \cdot |N_o| < |J|$ means that there are not enough resources.

For principle G6, we can state that g_1 is embedded in g_2 if

$$\{ n_r \mid \exists q, N_o (\langle n_r, q, N_o \rangle \in g_1.B) \} \subset \{ n_r \mid \exists q, N_o (\langle n_r, q, N_o \rangle \in g_2.B) \}$$

$$\{ n_a \mid \exists n_r, n_o (\langle n_a, n_r, n_o \rangle \in g_1.J) \} \subset \{ n_a \mid \exists n_r, n_o (\langle n_a, n_r, n_o \rangle \in g_2.J) \}$$

For G7, we can state that g_1 is overlapped with g_2 if

$$\{n_r \mid \exists q, N_o (<n_r, q, N_o > \in g_1.e.B)\} \cap \{n_r \mid \exists q, N_o (<n_r, q, N_o > \in g_2.e.B)\} \neq \phi$$

$$\{n_a \mid \exists n_r, n_o (<n_a, n_r, n_o > \in g_1.J)\} \cap \{n_a \mid \exists n_r, n_o (<n_a, n_r, n_o > \in g_2.J)\} \neq \phi$$

For G8, we can state that g is public if g is in all the roles' object sets ($\forall r(r \in R \rightarrow g.n \in r.N_o)$) and g is private if g is in only some roles' object sets ($\exists r(r \in R, g.n \notin r.N_o)$).

The computer science department of Nipissing University is a group, where:

- n = CS department of Nipissing University;
- e = the computer science department environment; and
- J includes all the <agent, role, object> tuples expressing its members, their roles and their accessible offices.

The above definition of a group states this fact that without the users' participation, no collaboration would be performed.

We use g to express a specific group and \mathcal{G} the set of all groups. The relationship between an agent and a group is similar to that between an agent and a role.

After a group is built, users can log in the system, join it by playing a role, and access a relevant object, and interacts with each other. To support a group to work is basically the routine function of a collaborative system. In this definition, for group g , role r , agent a and an object o , we have such a relationship:

$$a.n \in \{b \mid <b, r, o > \in g.J\} \rightarrow a.n \in r.N_o.$$

i.e., joining a group means playing a role, but not vice versa. Therefore, a role is independent of groups.

Now, a role-based multi-agent system Σ can be described as an 8-tuple $\Sigma ::= \langle C, O, \mathcal{A}, \mathcal{M}, \mathcal{R}, \mathcal{E}, \mathcal{G}, s_0 \rangle$, where

- C is a set of classes;
- O is a set of objects;
- \mathcal{A} is a set of agents;
- \mathcal{M} is a set of messages;
- \mathcal{R} is a set of roles;
- \mathcal{E} is a set of environments;
- \mathcal{G} is a set of groups; and
- s_0 is the initial state of the system.

With the above revised E-CARGO model, we can restate what Shakespere (As You Like It, Act II, Scene 7) said with the above notation: "All the world (C, O) is a stage (\mathcal{E}), and all the men and women merely players (\mathcal{A}); they all have their exits and entrances (Σ, \mathcal{G}, s_0); and one man in his time plays many parts (\mathcal{R})."

ARCHITECTURE OF RMAS

We can specify an agent's role using two components: the service interface, including incoming messages, and the request interface, including outgoing messages (Zhu and Zhou, 2006a). The human icon in Fig. 1 can be an agent, object, group or system. Hence, the roles applied in an agent system should be concerned with two aspects of roles: responsibilities and rights, because a player (an agent, or an object) should perform the service upon request and send out messages to seek other players' services. With the revised E-CARGO model, we know that every group has an environment in which agents play roles.

With roles, to form a group of agents who collaborate to complete a defined task, we suggest the following steps:

- Define roles required by the task (r);
- Connect the roles with structures (e, c, o);
- Design agents based on roles (a); and
- Release agents to play roles in the group (g).

To accomplish the above tasks, we need to provide tools as follows:

- Role specification;
- Role registration;
- Role match;
- Agent release; and
- Agent migration.

To develop an RMAS, the main tasks are specifying roles and the relationships among them, specifying role players and assigning roles to them. In the system design, roles are created, specified and connected into a net, called role net (Fig. 2). A system architecture, in fact, is an environment e of the E-CARGO model.

To provide a concrete system, we need to obtain the architecture with roles and role relationships, create agents that can play roles, assign agents with roles, and then allow agents to play roles. Hence, to develop an RMAS, we have the following steps: architecture design, agent implementation, and system integration (Zhu, 2006b).

Architecture design

In this step, the major task is to build role nets (Fig. 2). A role net is a blueprint of agent collaboration and is composed of roles, requests and services.

The basic procedure is as follows:

- 1) Identify roles. Analysts extract roles from the problem descriptions.
- 2) Specify roles. Designers describe the incoming and outgoing messages for the roles.
- 3) Specify the role relationships. Designers describe the relationships among the roles, such as classification, promotion, request/service, and conflict.

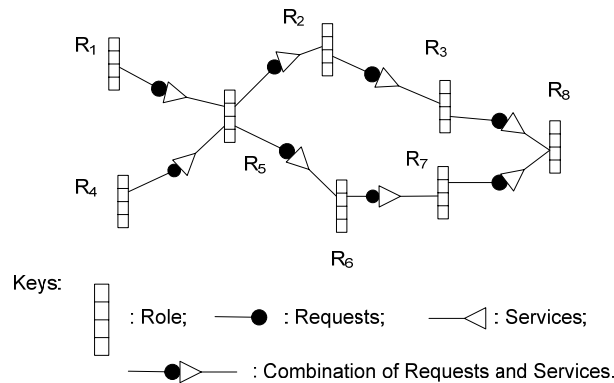


Fig. 2. Role Net: Architecture Design

Agent implementation

The major tasks in this step are to design and implement qualified role players, i.e., agents (Fig. 3).

RMAS implementers are mainly concerned with implementing agents. An agent might be a machine, a computer, a robot, a hardware component such as a sensor, or software component such as a process.

There are two possible explanations as to why an agent does not work: one is that the designers have not specified enough requests (or rights) for the roles; or the agent developers have not completely applied the specified requests (or rights). Designers should concentrate

on specifying roles and the relevant requests and services. They need only care about high-level role specifications and role request/service relationships. That is, which roles should be in a system; which rights roles should possess and which services roles should provide. Agent developers will work primarily on how to implement the services with the provided requests.

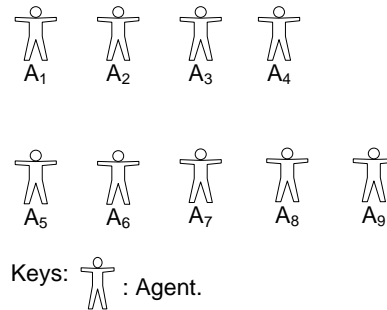


Fig. 3. Agents: Implementation of Role Players

System integration

The major task of system integration is to have agents play roles (Fig. 4), where “active roles” means that the relevant agents are qualified to play the relevant roles and “current roles” means the agents are currently playing the roles. They are discussed in detail in Section 6. Agents and roles are combined together to make a whole intelligent system executable.

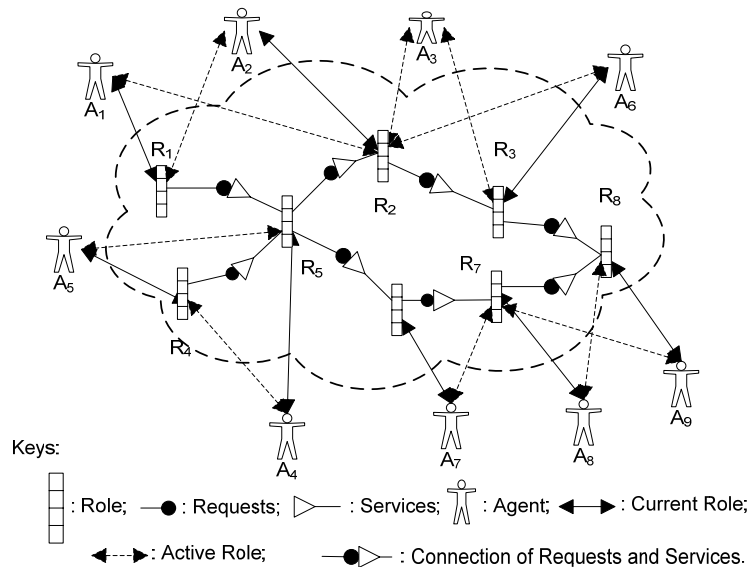


Fig. 4. System Integration: assigning roles to agents

A multi-agent system can be built by integration, i.e., assigning agents with roles, or having agents play roles. This step builds a deliverable and workable intelligent system (Fig. 4). At this step, agents are put onto the role nets, and match a role or roles to play. If each role has a relevant agent to play it and the agent can provide enough space and speed required by that role, the system is completed and workable.

Evidently, the above three steps can be done by specialists such as designers, agent providers, and system integrators who possess differing levels of experience and training. The integration step shows significant scaling capabilities based on assignment of sufficient agents to perform a single role. The efficiencies of agents as supplied by different agent

providers may vary. Based on these differences, the managers of a development team must have concrete evaluation criteria for these providers.

From Figs. 2-4, we find that a system is designed when roles are designed and specified. System construction involves the design of agents that are qualified to play roles then arrange for agents to play those roles.

From Figs. 2 and 3, we know that software design establishes roles and their relationships while system implementation calls agents and have them play roles.

Based on the above discussion, we find that the specialization of designers and that of agent providers are totally different. Designers are experts in role specification, role relationship analysis and role structure design. Agent providers are experts from different professions with special skills in the creation and provision of agents qualified to play roles as specified by designers. They require different consideration, expertise, and knowledge.

This specialization really separates designers and agent providers. This will lead to the real separation between system design and system implementation.

We also need other experienced specialists to match agents and roles. They understand agents and roles. They are match-makers for roles and agents. They are the bridges between design and implementation. Match-making is their major tasks. These specialists are a totally new type of system developer.

AGENT EVOLUTION IN RMAS

To build RMAS, we need to take roles as nodes in a distributed structure or architecture. Agents will be placed on it to do jobs. An agent can be an organization, group, person, system, machine (computer), or component of a machine.

Agents in a system work actively and collaboratively toward the common goal of the system. Introduction of a new agent to the system is done for a specific reason. Role performance is the reason why agents are created, modified and transferred. An agent should be able to adopt appropriate roles as demonstration of its ability and transfer roles to demonstrate mobility. The evolution of agents is based on their adaptabilities (Hayes-Roth, 1999). Agents are required to adapt five aspects of an intelligent system: perception strategy, control mode, reasoning tasks, reasoning methods, and meta-control strategy. Roles are a great tool to describe the evolution of agents in an intelligent system and improve their adaptability in the system. Past, active, current and future roles can well express the evolution of agents.

Future roles

In our society, people strive to be a great person within a great group. This situation was described early in 1970s by Maslow's hierarchy of needs (Maslow, 1970). Every person has a goal in a society. A goal is a controlling or guiding state that determines the action of search and selection and qualifies a person's success or failure.

In a society, roles are organized in a ladder mode to encourage people to pursue higher positions and obtain increased respect. These ladder modes can be seen almost everywhere. In a university, there is a ladder from assistant professor to full professor. In a software company, there is a ladder from programmer to senior programmer. In an army, there is a ladder from the lowest rank, recruit, to the highest rank, general. All of these ladders aim at encouraging people to work in order to obtain progressive promotion. Fig. 5 indicates examples of role hierarchy (or a ladder). A role hierarchy is actually a partial relation of roles, say, $\langle \mathcal{R} \rangle$, where \mathcal{R} is as defined in Section 4 and $r_1 < r_2$ means r_1 is a sub-role of r_2 and r_2 is a super-role of r_1 . For example, in Fig. 5, *Assistant Professor* \langle *Associate Professor*, *Full Professor* \langle *Distinguished Professor*, and *Research Member* \langle *Research Fellow*.

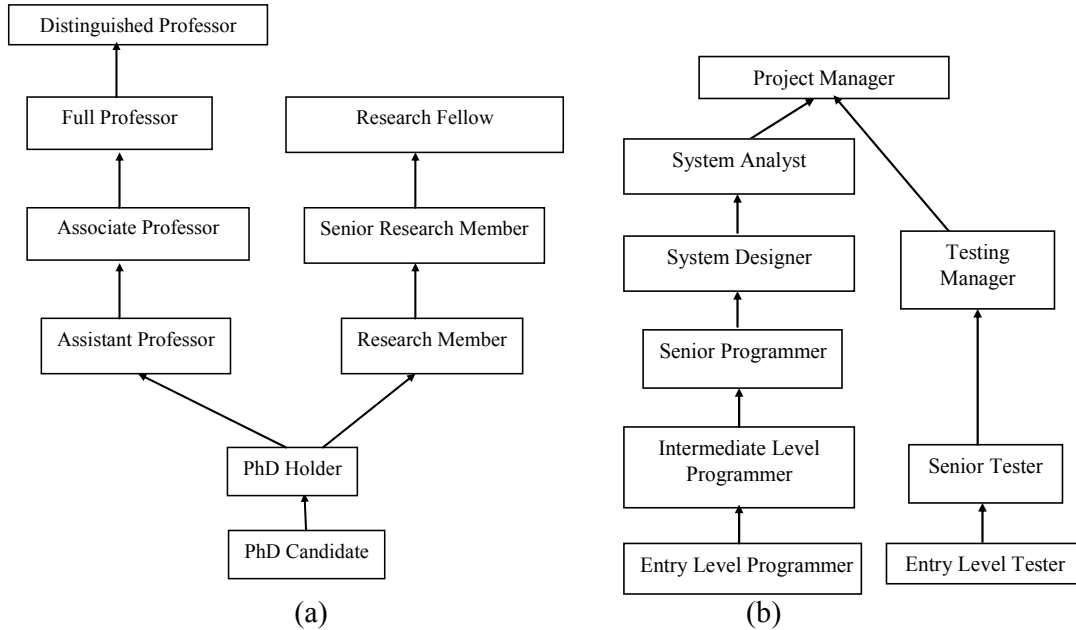


Fig. 5 Examples of role ladders for (a) a researcher and (b) a software developer.

When an agent lives in a role net, it tries to achieve its *future roles* through hard work. By “future, we mean the roles an agent hopes to play in the future. To be qualified to play the future role, the agent must collect credits by serving other agents in the system. Therefore, roles can be taken as goals for agents in a role-based agent system. Gradually, agents approach their goals.

The current roles and active roles

From daily lives, we know that a person may play many roles during a period. For example, a person might be a professor, technical consultant and project manager in the same year. To express this situation while obeying the principle “a player plays only one role at a time” (Zhu and Zhou, 2006a), we need to differentiate the concepts of *active roles* (\mathcal{R}_p in E-CARGO) and *the current role* (r_c in E-CARGO). By “active roles”, we mean the player still responds to the messages relevant to these roles but with delayed or scheduled responses. Sometimes, an active role should be transferred to the current role to respond to the messages. By “current”, we mean the agent is currently playing this role, i.e., it directly responds to the messages relevant to this role. These concepts are used to express the current state of an agent. Active roles are the roles a player is holding and they are ready to respond to messages. The current role is the role that can immediately respond to messages coming to it. That is to say, a role player can hold many active roles at the same time but it holds only one current role at a time. By holding only one current role, we can avoid role-role conflicts (Nyanchama, 1999). Active roles can also be used to express the meaningfulness of role transition, i.e., changing the current role from one active role to another.

Past roles

To express an agent’s dynamic and evolving situation, we must consider the passage of time. To completely express a live agent, we need to track past, current, active and future roles. This requires tracking of an historical record. We are then able to answer a question such as: What roles did an agent play in the past? To accomplish this task, we need to introduce the concept of *past roles*. By “past roles”, we mean those already performed by an agent then discarded. Fig. 6 shows the roles and the time segment used by an agent in their performance.

Suppose a person plays a student role again after playing a project manager role, should we create a new role instance or replay the original role instance? This will depend on actual requirements and the situation. If it is really a different one, say, a graduate student, we need to create a new role instance. If he/she really plays the same student role instance as before, he/she should replay the original role instance.

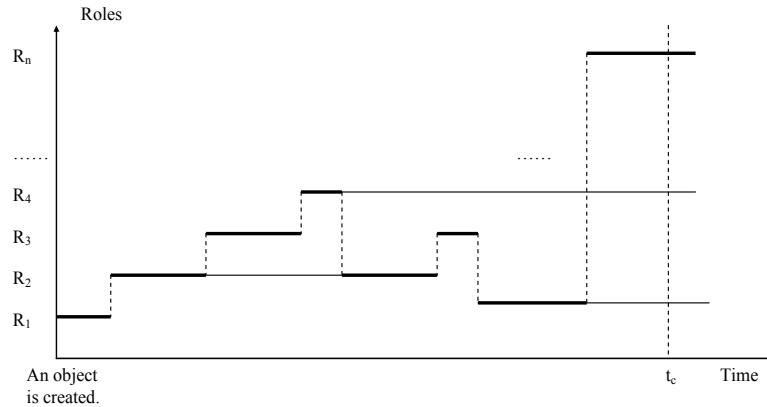


Fig. 6. The role play graph showing the roles an agent plays.

The horizontal axis stands for time and vertical one for roles. To view an agent's evolution, we can concentrate on its role transition along the time axis. In Fig. 6, the current roles are expressed by the bold line segments and the active roles by thin lines. A bold line segment expresses a past role. Suppose that the current time is t_c . The past roles of the agent in Fig. 6 were R_1 - R_3 , the active ones are R_1 , R_4 and R_n , and current one is R_n .

Role playing rules

To accommodate agent evolution, there should be a role engine that provides a method to attach an agent to a role and a method to find an agent to execute the message. These methods affect the promotion of the agents, i.e., to play more high level roles. They should check if the agent can be attached to the role and return true or false as a result and dispatch the messages evenly and without bias.

To determine if an agent is qualified to play a role, there are several criteria:

- The agent must provide methods to cover all the incoming messages for the role;
- The agent must collect enough credits;
- The agent must have the needed processing capacities;
- At least one role the agent is playing belongs to the sub-roles of this role if the role has sub-roles; and
- When a role is approved for an agent, the processing capacity $\langle e_t, e_s \rangle$ of the agent should be decreased. When an agent releases a role, $\langle e_t, e_s \rangle$ should be increased.

To manage credits, we assign a weight for each incoming message of a role. When we specify a role, we can assign incoming messages to a role and set the specific weight for the role. That is to say, the same incoming messages can have different weights for different roles. When an agent successfully executes a method relevant to an incoming message, it collects the weight of the message to increase its credits.

CASE STUDY

With the revised E-CARGO model discussed as above, we can define a soccer team with the 4-4-2 formulation as an environment as $e_1 = \{ \langle \text{goalie}, [1], f \rangle, \langle \text{defender}, [4], f \rangle, \langle \text{midfield}, [4], f \rangle, \langle \text{forward}, [2], f \rangle, \langle \text{goalie}, [1], f \rangle \}$, where, $f = \{ \text{the gate, the field, the ball} \}$. A 3-6-1 formulation can be described as $e_2 = \{ \langle \text{defender}, [3], f \rangle, \langle \text{midfield}, [6], f \rangle, \langle \text{forward}, [1], f \rangle, \langle \text{goalie}, [1], f \rangle \}$ (Zhu, 2006c).

If we specify the team in a more clear way, the environments are: $e_1 = \{ \langle \text{goalie}, [1], f \rangle, \langle \text{left-defender}, [1], f \rangle, \langle \text{right-defender}, [1], f \rangle, \langle \text{mid-defender}, [2], f \rangle, \langle \text{left-midfield}, [1], f \rangle, \langle \text{right-midfield}, [1], f \rangle, \langle \text{mid-midfield}, [2], f \rangle, \langle \text{left-forward}, [1], f \rangle, \langle \text{right-forward}, [1], f \rangle, \langle \text{goalie}, [1], f \rangle \}$. $e_2 = \{ \langle \text{left-defender}, [1], f \rangle, \langle \text{right-defender}, [1], f \rangle, \langle \text{mid-defender}, [1], f \rangle, \langle \text{left-midfield}, [2], f \rangle, \langle \text{right-midfield}, [2], f \rangle, \langle \text{mid-midfield}, [2], f \rangle, \langle \text{left-forward}, [1], f \rangle, \langle \text{goalie}, [1], f \rangle \}$.

The environment e_1 can be shown as in Fig. 7, where R_1 - R_4 are defender role instances, R_5 - R_8 are midfielder role instances, R_9 - R_{10} are forward role instances, and R_0 is the goalie role instance. Fig. 7 also shows some relationships among roles. For example, R_0 can request R_1 , R_2 , R_3 , and R_4 ; R_2 can request R_5 and R_6 ; and R_3 can request R_7 and R_8 .

Robots are designed and built to possess similar hardware abilities. They can sense, move, control a ball, and kick a ball.

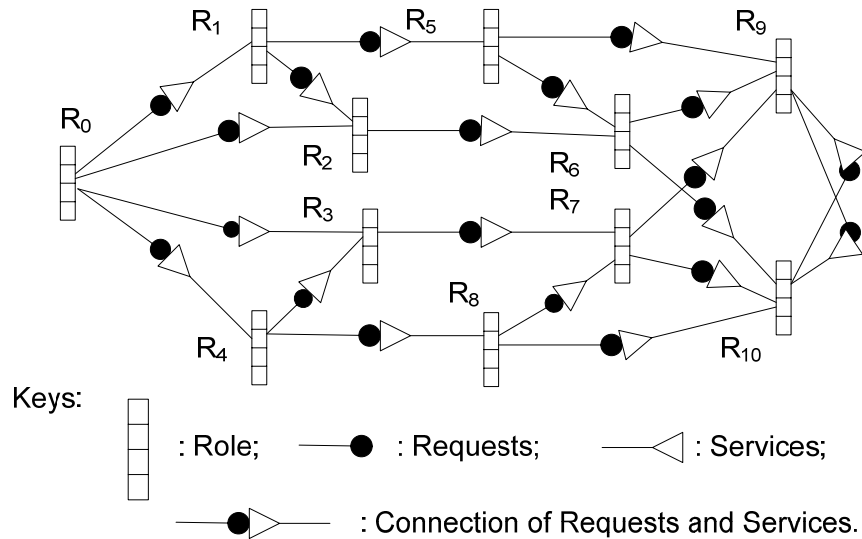


Fig. 7. An environment for a soccer team with the 4-4-2 formation.

For roles in a robot soccer team, a role can be specified as how to behave with Java (Zhu, 2007). For example, a forward role segment can be specified as:

```
public class forward extends Role {
    public forward(Robot nearest) //Constructor of the Forward class
    {
        //initialize the states of a forward at the beginning of a game
        distanceToNearestForward=140; //half of the field width
        distanceToGate=230; //less than the half of the field length
        distanceToGoal=30; //It depends on the power of the robot
        newrestForward=nearest;
        ballHoldState=false;
        blockedToGoal=true;
    }
    public void start(){
        while (true)
        {
            if (!ballHoldState) moveToBall();
            if ((ballHoldState)
                && (distanceToGate >= distanceToNearestForward)
                && (distanceToGoal >= distanceToGate)
                && (!blockedToGoal) )
                goal();
        }
    }
}
```

```

        if ((ballHoldState) && (!blockedToGoal))
            pass(newrestForward);
    }
}
private void moveToBall()
{    //movel to the ball;
}
private void goal()
{    //kick the ball to the gate;
}
private void pass(Robot r)
{    //pass the ball to the robot r;
}
}
private double distanceToNearestForward;
//in centimeters. The field size is 440 cm x 280 cm (Weiss and Reusch, 2005).
private double distanceToGate; //in centimeters
private double distanceToGoal; //in centimeters
private Robot newrestForward;
//an robot playing a Forward role. Robot is a class to describe all robots.
private boolean ballHoldState; //true means holding the ball and false not.
private boolean blockedToGoal; //true means blocked and false not.
}

```

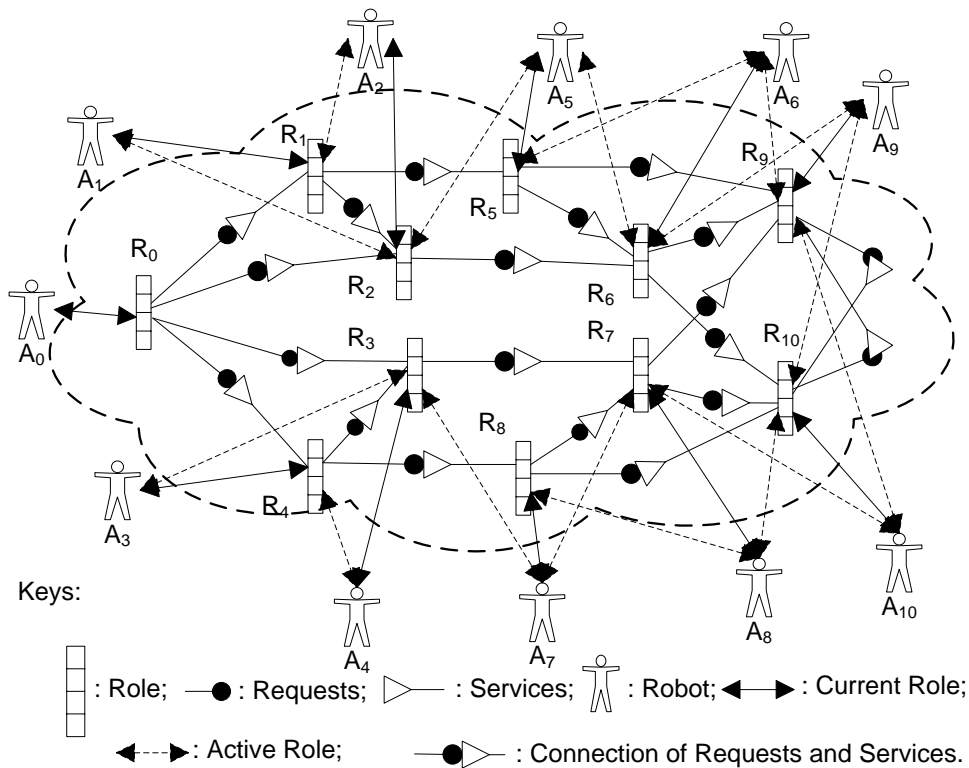


Fig. 8. A robot soccer team.

To organize a robot soccer team specified by Fig. 7, we can have 11 robots and assign them role instances (Fig. 8). Robots with limited power (electric and computing) and memories are often assigned to play similar roles as shown in Fig. 8. In such a situation, the role transfer for

each robot is limited. When robots are developed with enough power and memories, many or all roles can be assigned to every robot. Role transfers are triggered with events activated by the sensors and conditions predefined. Fig. 8 expresses that this formation is dynamic. It is initially 4-4-2. When attacking, it might evolve into 2-4-4. When blocking, it might be 4-6-0. This totally supports the dynamic role allocations for agent that are required in a robot agent team (Chaimowicz *et al.*, 2002; Stone and Veloso, 1999).

INFORMATION PERSONALIZATION WITH ROLES

Because of the vast quantity of information on the Internet, it is difficult for users to find information appropriate to their interests. It is necessary to filter irrelevant information from that presented to Internet users (Abidi and Zeng, 2006). Information personalization refers to the automatic adjustment of information content, structure, and presentation tailored to an individual user. It is obtained by reducing information overload and customizing information access. Information personalization is important in the development of a user-friendly interface to the Internet. It constitutes the mechanisms and technologies required to customize information access to end-users.

However, it is difficult to capture requirement specifications independent of particular personalization algorithms or techniques (Haase *et al.*, 2004, Perugini and Ramakrishnan, 2003; Ramakrishnan *et al.*, 2007). It is not practical for users to set their personal filtering preferences because many lack sufficient computer-literacy and they wish to limit their experience to clicking buttons or selecting menu items. Role-based approaches are a good trade-off between an easy end-user experience and the filtering out of irrelevant information. Roles enable users to easily set interface preferences relevant to their jobs. Through roles, common interfaces for specialized information can be designed thus filtering much irrelevant information.

Information personalization needs to address the following requirements (Abidi and Zeng, 2006):

- 1) The personalized information should be relevant to the interests of the user. The user may choose the degree of relevance to include either all or a partial list of topics of interest.
- 2) The personalized information should be factually consistent, i.e., the set of documents being presented to users should mutually satisfy the consistency constraint.
- 3) The information personalization process should attempt to find the largest set of consistent documents in terms of the coverage of topics defined by users.

Using roles, we can support 1) by indicating specific concerns; 2) by setting server agents to provide role-related information; and 3) by matching requests with agents performing the same role.

Based on the E-CARGO model, the scenario of role-based information personalization on the Internet is as follows:

- 1) Users use client computers to retrieve *information* stored in Internet server computers (client/server architecture);
- 2) All the information is stored and expressed in the format of *objects* and *classes*;
- 3) All the information is distributed across Internet *server* computers that form an *environment*;
- 4) In each *server* computer, there is one or many prescribed *roles*;
- 5) *Agents* are put into the computers to form a *group*;
- 6) Each *agent* plays one or many *roles* in the *group* to manage the information relevant to the *roles*;
- 7) In each *client* computer (personal computer), there is one *agent* that serves its user;
- 8) When a user logs in a *client* computer, an *agent* will present him/her *roles*;

- 9) Each time the user retrieves information, s/he plays a *role*;
- 10) It is much easier for users to set the criteria to search for information because each *role* helps him/her concentrate on a specific concern;
- 11) All the requests from users will be issued through their current *roles*;
- 12) The information request with its *role* identification is issued to an *agent* playing a specific *role broker* (Siegel, 1998);
- 13) This *agent* matches the *request* with the *service* of some *agents*;
- 14) The users obtain the *information* from the server *agent*.

In the above scenario, *agents* at the server computers can be specially designed to play roles for certain information efficiently and effectively; *roles* at the server computers can be used to shrink the information space to be searched and retrieved by a special request; *roles* at the client are used to concentrate on the information preferred by the user; and the *agents* at the client are used to collect the users' personal preferences and retrieval history. With this personal information, *roles* can be used to represent the user's preferences. The above process significantly increases the precision of the requested information.

FUTURE RESEARCH DIRECTIONS AND CONCLUSIONS

The human world can be characterized by three major statements:

- A society has people and roles;
- Roles are organized in structures; and
- People play roles.

To develop a role-based agent system, the main tasks are to specify roles and the relationships among roles, specify role players, and assign roles to their players. In the system design, roles are created, specified and connected into a net, called role net. System architecture, in fact, is an environment of the E-CARGO model. To provide a concrete system, we need to obtain the architecture with roles and role relationships, create agents that can play roles, assign agents with roles and then let agents play roles.

That is to say, to develop a role-based agent system, we have the following steps: architecture design, agent implementation, and system integration.

Roles are fundamental for multi-agent systems and information personalization. Using a role-based architecture, a multi-agent system can be designed by role and role relationship specification. We can consider a computer or a computer group as an agent in the networked systems. Every computer or computer group added to the network is to play a role or a group of roles in order to make the distributed system perform better.

A role-based architecture is a great guideline for distributed MAS builders and designers to plan and architect a distributed intelligent system. It is also an engine to help agents evolve to automatically join the architecture and cooperate with other agents to make MAS perform better. It is an exciting future job to design agent factories on networked multi-computer systems. They are built to meet the requirement of roles to solve a special practical problem. Role-based information personalization is an exciting topic to investigate. It inherently incorporates the requirement of information personalization.

The revised E-CARGO model has demonstrated the abstract mechanisms for role specification. We still need to provide concrete tools to specify, store, manage, transfer and apply roles in intelligent system development. Future work can be investigated in the following aspects:

- Design and implement role-based grid environment for agents;
- Design and implement role-based information personalization systems;
- Develop role-based robot collaboration, such as, robot soccer teams with better role assignment and role transfer strategies;
- Design role-based agent development tools; and

- Build simulation systems with roles and agents to help analyze human world community.

References:

1. Abidi, S. S. R., & Zeng, Y. (2006). Intelligent Information Personalization Leveraging Constraint Satisfaction and Association Rule Methods. in Lamontagne, L. & Marchand, M. (Eds.), *Proc. of Canadian AI 2006*, LNAI 4013, 134-145.
2. Ahn, H. J., Lee, H., & Park, S. J. (2003). A flexible agent system for change adaptation in supply chains. *Expert Systems with Applications*, 25(4), 603-618.
3. Alon, I. (2003). *Chinese Culture, Organizational Behavior, and International Business Management*, Westport: Conn Greenwood Publishing Group, 2003.
4. Ashforth, B. E. & Mahwah, N. J. (2001). *Role Transitions in Organizational Life: An Identity-Based Perspective*, Mahwah, NJ: Lawrence Erlbaum, 2001.
5. Becht, M., Gurzki, T., Klarman, J. & Muscholl, M. (1999). ROPE: Role Oriented Programming Environment for Multiagent Systems. *Proc. 4th IECIS International Conference of Cooperative Information Systems* (pp. 325-333), IEEE Computer Society, Washington, DC, USA.
6. Bellavista, P., Corradi, A., & Stefanelli, C. (2001). Mobile Agent Middleware for Mobile Computing. *IEEE Computer*, 34(3), 73-81.
7. Bostrom, R. P. (1980). Role Conflict and Ambiguity: Critical Variables in the MIS User-Designer Relationship. *Proc. of 17th Annual Computer Personnel Research Conference* (pp. 88-115), Miami, FL, USA.
8. Botha, R. A., & Eloff, J. H. P. (2001). Designing Role Hierarchies for Access Control in Workflow Systems. *Proc. of the 25th Annual International Computer Software and Applications Conference (COMPSAC'01)* (pp. 117-122), Chicago, Illinois, USA.
9. Bowling, M. & Veloso, M. (2002). Multiagent Learning Using a Variable Learning Rate. *Artificial Intelligence*, 136(2), 215-250.
10. Cabri, G., Ferrari, L., & Leonardi, L. (2005a). Injecting roles in Java Agents through Runtime Bytecode Manipulation. *IBM Systems Journal*, 44(1), 185-208.
11. Cabri, G., Ferrari, L., & Leonardi, L. (2005b). Supporting the Development of Multi-Agent Interactions via Roles. *The 6th International Workshop on Agent-Oriented Software Engineering (AOSE) at AAMAS 2005*(pp. 54-166), Utrecht, The Netherlands, July 2005, Springer LNCS vol. 3950.
12. Cao, L.B., Zhang, C.Q., Dai, R.W. (2005a). The OSOAD Methodology for Open Complex Agent Systems. *Int. J. of Intelligent Control and Systems*, 10(4), 277-285.
13. Cao, L.B., Zhang, C.Q., Dai, R.W. (2005b). Organization-Oriented Analysis of Open Complex Agent Systems. *Int. J. of Intelligent Control and Systems*, 10(2), 114-122.
14. Castelfranchi, C. (1998). Modeling Social Action for AI Agents. *Artificial Intelligence*, 103, 157-182.
15. Chaimowicz, L., Campos, M. F. M., & Kumar, R.V. (2002). Dynamic Role Assignment for Cooperative Robots. *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA '02)* (pp. 293-298), Washington, DC, USA.
16. Chockler, G.V., Keidar, I., & Vitenberg, I. (2001). Group Communication Specifications: A Comprehensive Study. *ACM Computing Surveys*, 33(4), 427-469.
17. Coulouris, G., Dollimore, J., & Kindberg, T. (2005), *Distributed Systems: Concepts and Design (4th ed.)*, Reading, MA: Addison-Wesley, 2005.
18. Cristiano, C. (1998). Modelling social action for AI agents. *Artificial Intelligence*, 103(1-2), 157-182.
19. Cyert, R. M., & MacCrimmon, K. R. (1968). Organizations. In G. Lindzey, & E. Aronson (Ed.), *The Handbook of Social Psychology*(pp. 568-611), 1, Addison-Wesley.

20. Dafoulas, G. A., & Macaulay, L. A. (2001). Facilitating Group Formation and Role Allocation in Software Engineering Groups. *Proc. ACS/IEEE International Conference of Computer Systems and Applications* (pp. 352–359), Jun. 25–29, 2001, Beirut, Lebanon.
21. Esteva, M., Rodríguez-Aguilar, J.A., Sierra C., Garcia, P., & Arcos, J.L., (2001). On the Formal Specifications of Electronic Institutions. *Lecture Notes in Computer Science*, 1991, 126-147.
22. Etzioni, O., & Weld, D.S. (1995). Intelligent Agents on the Internet: Fact, Fiction, and Forecast. *IEEE Expert: Intelligent Systems and Their Applications*, 10(4), 44-49.
23. Ferber, J., Gutknecht, O., & Michel, F. (2004). From Agents to Organizations: an Organizational View of Multi-Agent Systems. in Giorgini, P., Müller, J., & Odell, J. (Eds.), *Agent-Oriented Software Engineering (AOSE) IV, Melbourne, July 2003, Lecture Notes on Computer Science*, vol. 2935, 214-230.
24. Genesereth, M.R., & Ketchpel, S.P. (1994). Software Agents. *Communications of the ACM*, 37(7), 48-55.
25. Green, S., Hurst, L., Nangle, B., Cunningham, P., Somers, F., & Evans, R. (1997). Software Agents: A Review. *Technical report TCD-CS-1997-06*, Trinity College Dublin.
26. Gruver, W. (2004). Technologies and Applications of Distributed Intelligent Systems. *IEEE MTT-Chapter Presentation*, Waterloo, Canada, 2004.
27. Hayes-Roth, B. (1995). An Architecture for Adaptive Intelligent Systems. *Artificial Intelligence*, 72(1-2), 329-365.
28. Haase, P., Ehrig, M., Hotho, A., & Schnizler, B. (2004). Personalized Information Access in a Bibliographic Peer-to-Peer System. *Proceedings of the AAAI Workshop on Semantic Web Personalization*, 2004. Retrieved March 22, 2007, from: <http://citeseer.ist.psu.edu/haase04personalized.html>.
29. Heylighen, F. (1999). Collective Intelligence and its Implementation on the Web. *Computational & Mathematical Organization Theory*, 5(3), 253-280.
30. Jennings, N., & Wooldridge, M. (1996). Software agents. *IEE Review*, 42(1), 17-20.
31. Jennings, N., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 42(1), 7-38.
32. Kay, A. (1993). The early history of Smalltalk. *Proc. of The 2nd ACM SIGPLAN conference on History of Programming Languages* (pp. 69-95), Cambridge, Massachusetts, United States.
33. Kephart, J.O., Hanson, J.E., & Greenwald, A.R. (2000). Dynamic pricing by software agents. *Computer Networks*.
34. Lavender, R. G., & Schmidt, D. C. (1996). Active Object: an Object Behavioral Pattern for Concurrent Programming. In J. O. Coplien, J. Vlissides, & N. Kerth (Ed.), *Pattern Languages of Program Design*, Boston, MA: Addison-Wesley.
35. Loudon, K.C. (2003). *Programming Languages: Principles and Practice (2nd ed.)*, Brooks/Cole.
36. Maes, P. (1994). Modeling Adaptive Autonomous Agents. *Artificial Life*, 1994, 1(1), 135 - 162.
37. Maslow, A. (1970). *Motivation and Personality (2nd ed.)*, Harper & Row, 1970.
38. Meyer, B. (1988). *Object-Oriented Software Construction*, NJ: Prentice-Hall.
39. Mills, A.J., & Simmons, A.M. (1999), *Reading Organization Theory*, Toronto, ON: Garamond Press, 1999.
40. Miner, J. B. (1992). *Industrial-Organizational Psychology*. New York: McGraw-Hill, 1992.
41. Nwana H.S. (1996). Software Agents: An overview. *Knowledge Engineering Review*, 11(3), 205-244.

42. Nwana, H.S., Lee, L., & Jennings, N.R. (1996). Coordination in Software Agent Systems. *BT Technology Journal*, 14(4), 79-89.
43. Nyanchama, M., & Osborn, S. (1999). The Role Graph Model and Conflict of Interest. *ACM Trans. on Information and System Security*, 2(1), 3-33.
44. Odell, J., Nodine, M., & Levy, R. (2005). A Metamodel for Agents, Roles, and Groups. In Odell, J., Giorgini, P., Müller, J. (Ed.), *Agent-Oriented Software Engineering (AOSE), Lecture Notes on Computer Science*, 3382, Springer, Berlin, 78-92.
45. Odell, J., Van Dyke Parunak, H., & Fleischer, M. (2003). The Role of Roles in Designing Effective Agent Organizations. In Garcia, A.; Lucena, C.; Zambonelli, F.; Omicini, A.; Castro, J. (Eds.), *Software Engineering for Large-Scale Multi-Agent Systems, Lecture Notes on Computer Science*, vol. 2603, Springer, Berlin, 27-38.
46. Ould, M. A. (1995), *Business Processes: Modeling and Analysis for Re-engineering and Improvement.*, New York: Wiley, 1995.
47. Papazoglou, M. P. (2001). Agent-Oriented Technology in Support of E-Business. *Communication of the ACM*, 44(4), 71-77.
48. Perugini, S., & Ramakrishnan, N. (2003). Personalizing Interactions with Information Systems. In Zekowitz, M. (Eds.), *Advances in Computers*, 57, Academic Press, 323-382.
49. Pham, V.A., & Karmouch, A. (1998). Mobile Software Agents: An Overview. *IEEE Communications Magazine*, July 1998, 26-37.
50. Ramakrishnan, N., Rosson, M.B., & Carroll, J.M. (2007). Explaining Scenarios for Information Personalization. *arXiv:cs/0111007v1 [cs.HC]*. Retrieved April 10, 2007, from: <http://arxiv.org/pdf/cs/0111007v1>.
51. Russell, D., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach (2nd Ed.)*, Pearson Education, Inc., Upper Saddle River, New Jersey.
52. Scherer, A.G., (2003). Modes of Explanation in Organization Theory. In H. Tsoukas, & C. Knudsen (Eds.), *The Oxford Handbook of Organizational Theory*, Oxford: Oxford University Press, 310-344.
53. Siegel, J. (1998). OMG Overview: CORBA and the OMA in Enterprise Computing. *Communications of the ACM*, 41(10), 37-43.
54. Singley, M. K., Singh, M., Fairweather, P., Farrell, R., & Swerling, S. (2000). Algebra Jam: Supporting Teamwork and Managing Roles in a Collaborative Learning Environment. *Proc. CSCW'00* (pp. 145-154), Philadelphia, PA, USA.
55. Stone, P., & Veloso, M. (1999). Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. *Artificial Intelligence*, 110, 241-273.
56. Tanenbaum, A.S., & van Steen, M. (2002), *Distributed Systems: Principles and Paradigms*, NJ: Prentice-Hall, 2002.
57. WebBoard™ (2006), *WebBoard Discussion Forum and Collaboration Software*. Retrieved March 10, 2007, from: <http://www.webboard.com/>.
58. Webber, A.B. (2003), *Modern Programming Languages*, Franklin, Beedle & Associates.
59. Weiss, N., & Reusch, B. (2005). Current and Future Trends and Challenges in Robot Soccer. R. Moreno Diaz *et al.* (Eds.), *EUROCAST 2005*, LNCS 3643, 559-564.
60. Wooldridge, M., & Jennings, N. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2), 115-152.
61. Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3), 285-312.
62. Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering Methodology*, 12(3). 317-370.

63. Zhu, H. (2005, October). Encourage Contributions by Roles. *Proc. of the IEEE International Conference on Systems, Man and Cybernetics* (pp. 1574-1579), Hawaii, USA.
64. Zhu, H. (2006a). Role Mechanisms in Collaborative Systems. *International Journal of Production Research*, 41(1), 181-193.
65. Zhu, H. (2006b). A Role-Based Architecture for Intelligent Agent Systems. *Proc. of International Workshop on Distributed Intelligent Systems* (pp. 354-359), Prague, Czech.
66. Zhu, H. (2006c). A Role-Based Approach to Robot Agent Team Design. *Proc. of IEEE International Conference on Systems, Man and Cybernetics* (pp. 4861–4866), Oct. 2006, Taipei, China.
67. Zhu, H. (2006d). Separating Design from Implementations: Role-Based Software Development. *Proc. of the 5th IEEE International Conference on Cognitive Informatics* (pp. 141-148), Beijing, China.
68. Zhu, H. (2007). *Role Playing Java Documents*. Retrieved April 10, 2007, from: <http://www.nipissingu.ca/faculty/haibinz/RolePlaying/> .
69. Zhu, H., & Zhou, M.C. (2003). Methodology First and Language Second: A Way to Teach Object-Oriented Programming. *Companion of the ACM International Conference of Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'03)*, California, USA, Oct., 140-147.
70. Zhu, H., & Zhou, M.C. (2006a). Role-Based Collaboration and its Kernel Mechanisms. *IEEE Trans. on Systems, Man and Cybernetics, Part C*, 36(4), 578-589.
71. Zhu, H., & Zhou, M.C. (2006b). Supporting Software Development with Roles. *IEEE Trans. on Systems, Man and Cybernetics, Part A*, 36(6), 1110-1123.

Additional Reading:

The readers who are interested in agent systems could refer to the publications as follows:

1. Jennings, N., & Wooldridge, M. (1996). Software agents. *IEE Review*, 42(1), 17-20.
2. Jennings, N., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 42(1), 7-38.

The above two papers provide a good overview of research and development activities in the field of autonomous agents and multi-agent systems. They discuss the key concepts and applications and highlight a range of open issues and future challenges.

3. Maes, P. (1994). Modeling Adaptive Autonomous Agents. *Artificial Life, 1994*, 1(1), 135 - 162.

This paper discusses autonomous agents from the point of view of animal behaviors. From this point of view, it extracts its main ideas, evaluates what contributions have been made so far and identifies its current limitations and open problems.

4. Nwana H.S. (1996). Software Agents: An overview. *Knowledge Engineering Review*, 11(3), 205-244.

This paper reviews software agents and contains some strong opinions that are not necessarily widely accepted by the agent community.

5. Nwana, H.S., Lee, L., & Jennings, N.R. (1996). Coordination in Software Agent Systems. *BT Technology Journal*, 14(4), 79-89.

This paper examines co-ordination in multi-agent systems and highlights the necessity for co-ordination in agent systems, overviews briefly various co-ordination techniques and presents some conclusions and challenges drawn from this literature.

6. Russell, D., & Norvig, P. (2003), *Artificial Intelligence: A Modern Approach (2nd Ed.)*, Pearson Education, Inc., Upper Saddle River, New Jersey.

This book is the current standard college text book on Artificial Intelligence. The authors present the subject of artificial intelligence (AI) in a unified manner using the concept of an intelligent agent to develop a common framework for problem solving. The theory and practice of intelligent agent design are given equal weight in the text.

7. Wooldridge, M., & Jennings, N. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2), 115-152.

The paper identifies and elaborates on the most important issues of intelligent agents. It provides a short review of applications of agent technology at that time.

Artificial intelligence has a long history of research. There would be a much longer list for a full list of recommended readings. The list here only helps readers understand the initiatives of applying roles in multi-agent systems. The above readings give good overviews of agents and agent (mobile or multi-agent) systems. Readers could understand the general principles and important challenges related with agents, agent systems and collaboration among agents.

The readers who are interested in roles in agent systems could refer to the publications reviewed in this chapter as follows:

8. Becht, M., Gurzki, T., Klarmann, J., & Muscholl M. (1999). ROPE: Role Oriented Programming Environment for Multiagent Systems. *Proc. of Fourth IECIS International Conference on Cooperative Information Systems* (pp.325-333), Sept. 1999.
9. Cabri, G., Ferrari, L., & Leonardi, L. (2005a). Injecting roles in Java Agents through Runtime Bytecode Manipulation. *IBM Systems Journal*, 44(1), 185-208.
10. Cabri, G., Ferrari, L., & Leonardi, L. (2005b). Supporting the Development of Multi-Agent Interactions via Roles. *The 6th International Workshop on Agent-Oriented Software Engineering (AOSE) at AAMAS 2005*(pp. 54-166), Utrecht, The Netherlands, July 2005, Springer LNCS vol. 3950.
11. Cao, L.B., Zhang, C.Q., Dai, R.W. (2005a). The OSOAD Methodology for Open Complex Agent Systems. *Int. J. of Intelligent Control and Systems*, 10(4), 277-285.
12. Cao, L.B., Zhang, C.Q., Dai, R.W. (2005b). Organization-Oriented Analysis of Open Complex Agent Systems. *Int. J. of Intelligent Control and Systems*, 10(2), 114-122.
13. Depke, R., Heckel, R., & Kuster, J. M. (2001). Roles in Agent-Oriented Modeling. *International Journal of Software Engineering and Knowledge Engineering*, 11(3), 281-302.
14. Ferber, J., Gutknecht, O., & Michel, F. (2004). From Agents to Organizations: an Organizational View of Multi-Agent Systems. in *Giorgini, P., Müller, J., & Odell, J. (Eds.), Agent-Oriented Software Engineering (AOSE) IV, Melbourne, July 2003, Lecture Notes on Computer Science*, vol. 2935, 214-230.
15. Odell, J., Nodine, M., & Levy, R. (2005), "A Metamodel for Agents, Roles, and Groups", in *Odell, J., Giorgini, P., Müller, J. (Ed.), Agent-Oriented Software Engineering (AOSE), Lecture Notes on Computer Science*, vol. 3382, Springer, Berlin, 78-92.
16. Odell, J., Van Dyke Parunak, H., & Fleischer, M. (2003), "The Role of Roles in Designing Effective Agent Organizations," in *Garcia, A.; Lucena, C.; Zambonelli, F.; Omicini, A.; Castro, J. (Eds.), Software Engineering for Large-Scale Multi-Agent Systems, Lecture Notes on Computer Science*, vol. 2603, Springer, Berlin, 27-38.
17. Partsakoulakis, I., & Vouros G. (2004). Roles in MAS: Managing the Complexity of Tasks and Environments. In *Wagner, T. (Ed.), An Application Science for Multi-Agent Systems*, Springer, 133-154.
18. Stone, P., & Veloso, M. (1999). Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. *Artificial Intelligence*, 110, 241-273.

19. Wooldridge, M.; Jennings, N. R., & Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3), 285-312.
20. Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering Methodology*, 12(3), 317-370.

Agent systems are currently the most active research field that applies role concepts and mechanisms. The above readings can activate readers to think and understand the importance of roles in agent systems. They also give readers a good overview of how roles have been applied in agent systems and what aspects of roles are currently emphasized. From these readings, readers will find that roles can be abstract like interfaces and concrete like processes. Roles have many aspects to consider in different usages.

The readers who are interested in the general ideas of information personalization could further refer the publications as follows:

21. Abidi, S. S. R., & Zeng, Y. (2006). Intelligent Information Personalization Leveraging Constraint Satisfaction and Association Rule Methods. in Lamontagne, L., & Marchand, M. (Eds.), *Proc. of Canadian AI 2006*, LNAI 4013, 134-145.
22. Haase, P., Ehrig, M., Hotho, A., & Schnizler, B. (2004). Personalized Information Access in a Bibliographic Peer-to-Peer System. Proceedings of the AAAI Workshop on Semantic Web Personalization, 2004. Retrieved March 22, 2007, from: <http://citeseer.ist.psu.edu/haase04personalized.html> .
23. Perugini, S., & Ramakrishnan, N. (2003). Personalizing Interactions with Information Systems. in *Zelkowitz, M. (Eds.), Advances in Computers*, 57, Academic Press, 323-382.
24. Ramakrishnan, N., Rosson, M.B., & Carroll, J.M. (2007). Explaining Scenarios for Information Personalization. *arXiv:cs/0111007v1 [cs.HC]*. Retrieved April 10, 2007, from: <http://arxiv.org/pdf/cs/0111007v1>.

Information personalization is a new frontier of information technology. The above readings can activate readers to think and understand the importance of roles in information personalization. They also give readers a good overview of the challenges of information personalization. From these readings, readers will find that roles discussed in this chapter are good potential ways to solve the issues from the basic requirement of information personalization.

The readers who are interested in role-based collaboration could further refer to the following publications:

25. Zhu, H. (2005, October). Encourage Contributions by Roles. *Proc. of the IEEE International Conference on Systems, Man and Cybernetics* (pp. 1574-1579), Hawaii, USA.
26. Zhu, H. (2006a). Role Mechanisms in Collaborative Systems. *International Journal of Production Research*, 41(1), 181-193.
27. Zhu, H. (2006b). A Role-Based Architecture for Intelligent Agent Systems. *Proc. of International Workshop on Distributed Intelligent Systems* (pp. 354-359), Prague, Czech.
28. Zhu, H. (2006c). A Role-Based Approach to Robot Agent Team Design. *Proc. of IEEE International Conference on Systems, Man and Cybernetics* (pp. 4861—4866), Oct. 2006, Taipei, China.
29. Zhu, H. (2006d). Separating Design from Implementations: Role-Based Software Development. *Proc. of the 5th IEEE International Conference on Cognitive Informatics* (pp. 141-148), Beijing, China.

30. Zhu, H., & Zhou, M.C. (2006a). Role-Based Collaboration and its Kernel Mechanisms. *IEEE Trans. on Systems, Man and Cybernetics, Part C*, 36(4), 578-589.
31. Zhu, H., & Zhou, M.C. (2006b). Supporting Software Development with Roles. *IEEE Trans. on Systems, Man and Cybernetics, Part A*, 36(6), 1110-1123.

Role-based collaboration is an emerging technology that can be applied to many fields. The above papers discuss systematically the principles of role-based collaboration (Zhu and Zhou, 2006a), and present some possible application of role-based methodology (Zhu, 2005; Zhu, 2006a-d; Zhu and Zhou, 2006b). They are helpful for people to understand role-based collaboration and apply role-based methodology to their particular applications.