

# A ROLE-BASED ARCHITECTURE FOR INTELLIGENT AGENT SYSTEMS

Haibin Zhu, *Senior Member, IEEE*

*Department of Computer Science and Mathematics, Nipissing University,  
100 College Dr., North Bay, ON, Canada, P1B 8L7*

[haibinz@nipissingu.ca](mailto:haibinz@nipissingu.ca)

## Abstract

*This paper explores the properties of intelligent agent systems, introduces the model of role-based collaboration E-CARGO, and proposes that roles can be taken as an underlying mechanism to build intelligent agent systems by describing the process of developing agent systems and the agent dynamics in role-based agent systems. Finally, it points out the research topics of role-based distributed intelligent systems.*

**Keywords:** *Roles, Role-Based, Architecture, Agent, Intelligent Agent Systems*

## 1. Introduction

Artificial Intelligence is the discipline aimed at understanding intelligent beings by constructing intelligent systems [2]. From the point of view of behaviorist, intelligent systems are those systems that can simulate human beings' work that requires intelligence including logic inferring, problem solving, deduction and induction. A distributed system is composed many computers inter-connected with networks which cooperate and coordinate to accomplish one common task [3]. Distributed intelligent systems are intelligent systems built on a distributed computer system. They are based on the use of cooperative agents and organized in hardware or software components. In the system, each agent independently handles a small set of specialized tasks and cooperates to achieve the system-level goals and a high degree of flexibility [6].

From these definitions, we know that distributed intelligent systems are simulations of people societies or virtual societies. To simulate real societies, we need to understand the nature of our societies.

A well-organized society should encourage the members of the society to contribute. With the same requirement, distributed intelligent systems as virtual societies should also create a healthy platform or environment for virtual participants to contribute and work in the virtual societies. We can take agents as virtual persons in intelligent systems.

A good political system creates a good, steady, harmonious, and flourishing society. A good architecture

creates a strong and beautiful building. Therefore, a good distributed intelligent system requires a good system architecture. Role-based architecture is one candidate of such architectures.

From organizational and social psychological theory, an organization is composed of three key elements: participants, goals and roles [5]. Where, participants represent the people working in the organization; goals represent the future achievements all the participants jointly pursue; and roles represent the ways for participants to interact and achieve the goals. If we map a distributed intelligent system to an organization, we can quickly get what we need in it: agents, goals, and roles.

A goal is a mental representation of a process which expresses an agent's internal anticipation and regulations [2]. Therefore, it is meaningful to use roles to express a participant's goal in an organization [17]. Furthermore, we can even say that roles, role relationships, role players (agents) and role playing are all the aspects for distributed intelligent systems.

Therefore, the introduction of roles into system design has many potential benefits such as encouraging people's contributions to collaboration [17]; distributing tasks evenly; promoting traceability; facilitating reuse, and simplifying agents' complexity.

Roles are commonly applied concepts in many fields, such as sociology, psychology, social psychology, behavioral science, management, and drama. Roles can be used to map the people's natural organizations, task distributions, and system analysis, system design and system construction.

We believe that distributed intelligent systems are aiming to obtain collective intelligence [8] by organizing a large quantity of simple agents. Therefore, the author proposes role-based intelligent systems in order to apply roles completely and fundamentally into intelligent systems development.

In this paper, the author proposes a role-based architecture which provides a new methodology to improve the development of agent systems including system development and system design.

This paper is arranged as follows. Section 2 reviews the major concepts of agents; Section 3 briefly describes

the E-CARGO model; Section 4 demonstrates a role-based architecture for agent systems; Section 5 discusses the agent dynamics expressed by roles; Section 6 discusses the challenges in role-based architecture; and Section 7 concludes the paper and proposes the topics for future research.

## 2. Agents and Distributed Intelligent Systems

Multiagent systems are becoming more relevant to artificial intelligence [1]. Distributed intelligent systems are built with cooperative agents [6]. We need to understand the concept of an agent at first. The agent concept evolves from objects. It is a combination of object-orientation and artificial intelligence (AI). The AI community claims intelligence as a natural quality of agents and uses the traditional symbolic representation to describe agents.

Maes [12] defines agents as computational systems that inhabit some 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 and Jennings [16] define agents as hardware-based or (more usually) 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 some 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, the INTERNET, 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 the initiative.

Normal objects can be thought of as passive ones, because they wait for a message before performing an operation. Then, once invoked, they execute their method and go back to "sleep" until the next message. A trend in many systems now is to design objects that react to events in their environment, as well as be proactive. In pattern languages and Unified Modeling Language (UML), these are known as *active objects* [9]; in the agent community, they are known as *agents*.

Therefore, besides the characteristics of objects, an agent should also have the following characteristics:

- Active, i.e., an agent might act by its internal states. Note that an object in its conventional meaning can

only respond to the messages sent to it even though we acclaim that everything is an object;

- Autonomous, i.e., an agent is not controlled directly by people; and
- Collaborative, i.e., agents need to collaborative with others to accomplish a complex task.

Every agent is responsible for accomplishing a certain task. An agent can be considered as a self-contained object of some class, which does not belong to the problem domain although it does involve itself with that domain or environment. The environment exists before agents are created. A system works even without agents. The agent is an optional object with its own environment. The user of an agent will initiate it with some input and the agent becomes the object which is in charge of using the service of one or more objects from the problem domain or the system, in some sequence, to obtain a result.

Agents support a natural merge of object-orientation and knowledge-based technology [14]. They can facilitate the incorporation of reasoning capabilities within the application logic. They permit the inclusion of learning and self-improvement capabilities at both infrastructure and application levels. Unlike objects, agents can participate in high-level dialogues through the use of interaction protocols in conjunction with built-in organization knowledge. In many cases, the need for communication is greatly reduced, as within these high-level dialogues, the complex packets of procedural and declarative knowledge, as well as state information, may be exchanged in the form of objects. We call these objects as mobile objects because they can migrate among the agents. In addition, agent technology can help address serious technological challenges such as concerns about effective searching, security and privacy, and effective use of interoperability between diverse processes and diverse information required to achieve cooperation on distributed computers.

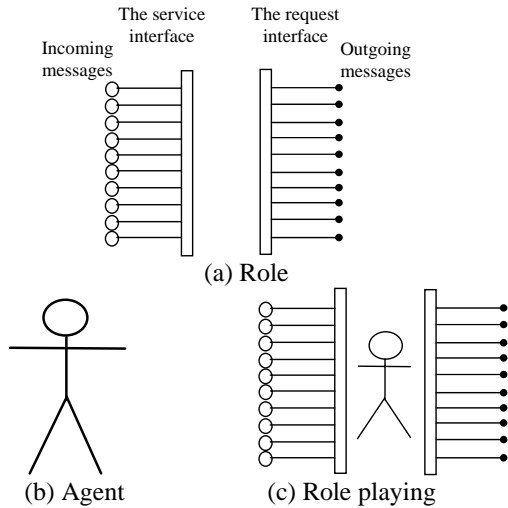
Agent designers must have an accurate and complete representation of the knowledge that will be used and generated by the system at the time the agents are designed. The artificial intelligence community has developed several approaches to represent this body of knowledge such as logic programming with Prolog, and Lambda Calculus with Lisp and ML [10, 15].

Agent designing is a complex task. Designers need to simulate intelligent procedures of people in an agent. Searching and retrieving are a way to express intelligence. Roles can be used to simplify the complexity of agent design, because roles separate concerns of an agent into different aspects. Based on the theory of searching and retrieving, this separation greatly shrinks the space for searching and will significantly increasing the efficiency for agents to respond messages or events relevant to a specific role.

### 3. E-CARGO Model

We can specify a person's role with two parts: the service interface including incoming messages, and the request interface including outgoing messages [18, 19]. In fact, the human icon in Fig. 1(b) and (c) can be replaced by an agent, object, group or system. Hence, the roles applied in an information system should be concerned about the two aspects of roles: responsibilities and rights, because a player (a person, an agent, or an object) should send out messages to invoke other players' services. In this paper, we concentrate on that an agent or a person plays a role.

To support role-based collaboration, we developed the E-CARGO model [18, 19] and revise it to encourage people to contribute in collaboration [17]. A collaborative system  $\Sigma$  can be described as a 9-tuple  $\Sigma ::= \langle C, O, \mathcal{A}, \mathcal{M}, \mathcal{R}, \mathcal{E}, \mathcal{G}, s_0, \mathcal{H} \rangle$ , 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,  $s_0$  is the initial state of a collaborative system, and  $\mathcal{H}$  is a set of users.



**Fig. 1. A role as a wrapper of a person in a working environment**

An *object* is everything in a system that occupies a memory cell or cells. It can be accessed and its data or status can be changed by operations on it.

A *class* is a template for a group of similar objects. It describes the operations on the group of objects and specifies the data structure of these objects.

An *agent* is a special object that represents a user involved in collaboration. It is defined as  $a ::= \langle n, c_a, s, \mathcal{N}_r, \mathcal{N}_g, e_v, e_s, w, u \rangle$ , where,  $c_a$  is a special class that describes the common properties of users,  $n$  is the identification or name of the agent,  $s$  is the set of properties of the agent,  $\mathcal{N}_r$  means a set of identifications of roles the agent is playing, and  $\mathcal{N}_g$  means a set of identifications of groups the agent

belongs to;  $e_v$  and  $e_s$  are used to express the ability,  $w$  the credit and  $u$  the workload.

A *message* is defined as  $m ::= \langle n, v, l, \mathcal{P}, t, w \rangle$  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  $\mathcal{P} \subset O$ ,  $t$  is a tag that expresses any, some or all message,  $w$  is the weight for an agent to collect its credit for promotion, i.e., if an agent or its human user responds this message the agent will get the weight and accumulate it to its credit.

A *role*  $r ::= \langle n, I, \mathcal{N}_a, \mathcal{N}_o, e_v, e_s, \mathcal{R}_m, \mathcal{R}_s, w \rangle$  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,  $\mathcal{M}_{out}$  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; and  $\mathcal{N}_o$  is a set of identifications of objects including classes, environments, roles, and groups that can be accessed by the agents playing this role;  $e_v$  and  $e_s$  are used to express the ability requirement,  $\mathcal{R}_m$  the super roles,  $\mathcal{R}_s$  the subordinate roles, and  $w$  the credit requirement.

An *environment* expresses a structure to build a group. It is specified by roles, the objects accessed by the roles and the cardinalities of agents playing roles.

A *group* is a set of agents that are working on an environment, i.e., a set of agents assigned with roles in the relevant environment.

With the E-CARGO model, if we consider *human users* as agent developers and *agents* as autonomous, the model becomes a model of role-based agent systems. Every group has an environment in which agents play roles.

### 4. Role-Based Architecture for Distributed Intelligent Systems

Our versatile world can be outlined with three major aspects:

- A society is formulated with roles;
- Roles are organized in structures; and
- People play roles.

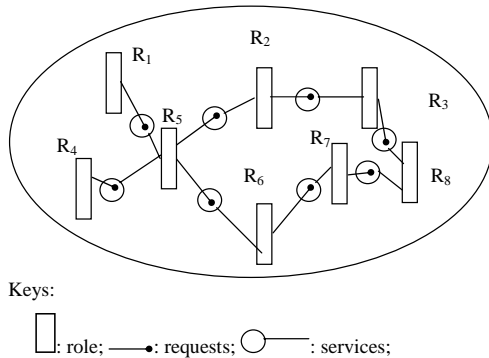
To develop a role-based system, the main tasks are specifying roles and the relationships among roles, specifying role players and assign roles to role players. In the system design, roles are created, specified and connected into a net, called role net (Fig. 2). A system architecture, in fact, is an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

To provide a concrete system, we need to obtain an 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 intelligent system, we have the following steps: architecture design, agent implementation, and system integration.

### 4.1 Architecture design

In this step, the major tasks are to build role nets (Fig. 2). A role net is composed of roles, request handle and service handle.



Keys:  
 □: role; —●: requests; ○—: services;

Fig. 2. Role Net: Architecture Design

The basic procedure is as follows:

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

### 4.2 Agent implementation

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

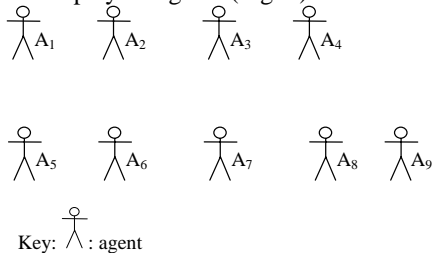


Fig. 3. Agents: Implementation of Role Players

In a role-based distributed intelligent system, 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 a software component such as a process.

There are two possible explanations for an agent that does not work: one is that the designers have not specified enough requests (or rights) for the roles; or the agent

developers did not completely apply the specified requests (rights). The designers should concentrate on specifying roles and the relevant requests and services, they only need to care about the high-level role specifications and the role requests/services relationships. That is, what roles should be in a system; what rights roles should require; what services roles should provide. The agent developers will mainly work on how to implement the services with the provided requests.

### 4.3 System integration

The major task of system integration is to have agents play roles (Fig. 4). Agents and roles are combined together to make a whole intelligent system executable.

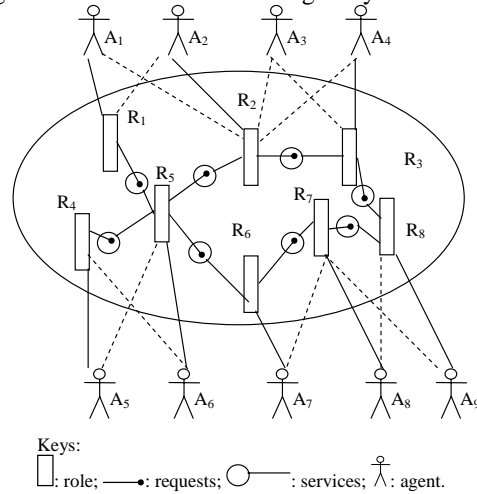


Fig. 4. Assign Agents with Roles-System Integration

An intelligent system can be built by integration, i.e., assigning agents with roles, or having agents play roles. This step will build the deliverable and workable intelligent system (Fig. 4). At this step, agents are put onto the role nets, 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 of 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 with different experiences and trainings. The integration step shows a lot of scaling capabilities based on how many agents play a single role. The efficiencies of different agents provided by different agent providers may be different. Based on these differences, managers of the development team have concrete evaluation criteria for agent providers. From Fig.2-4, we find that a system is designed when roles are designed and specified. System construction is to design agents that are qualified to play the roles and arrange agents to play roles.

With Fig. 2 and 3, we know that software design is to design roles, and the relationships among the roles, and system implementation is to make agents and have the agents 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 of role specifications, role relationship analysis and role structure design. Agent providers are experts with special skills in different professions to create and provide agents that are qualified to play roles specified by designers. They are different at considerations, 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 specialists who are experts to match agents and roles. They know the agents and know the basic specification of roles. They are match-makers of roles and agents. They are the bridges between designs and implementation. Match-making is their major tasks for them. These specialists are totally new types of system builders.

## 5. Agent Dynamics with Roles

The dynamics of agents are based on their adaptabilities [7]. Agents are required to adapt five aspects of an intelligent system: perception strategy, control mode, reasoning tasks, reasoning methods, and meta-control Strategy. To describe the dynamics of agents in an intelligent system and help agents' adaptability in a system, roles are a wonderful tool. Past Roles, Active Roles, Current Roles and Future Roles are enough and qualified to express the dynamics of agents.

### 5.1 Future roles

In our social life, people work hard to try to be a great person in a great group. This situation was described early in 1970s by Maslow's hierarchy [11] of needs. Every person has a goal in a society. A goal is a controlling or guiding action by means of repeated tests of the action's expected or actual results, determines the action of search and selection and qualify a person's success or failure [2]. In a society, roles are organized in a ladder mode to encourage people to pursue higher positions and obtain enough respects. 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 army, there is a ladder from the lowest rank recruit to the highest rank marshal. All the above ladders are aiming to encourage people to work in order to promote continually. Therefore, roles are an evident dynamics for people in our societies.

When an agent lives in a role net, it tries to achieve its *future roles* by working hard. 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, the agents approach their goals.

### 5.2 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, a technical consultant and a project manager in the same year. To express this situation and be consistent with the principle "a player plays only one role at a time" [18, 19], we need to differentiate the concepts of *active roles* and *current roles*. By "active", we mean the player still responds to the messages relevant to these roles. 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. They are used to answer such questions: What are its active roles? What is its current role?

Active roles are the roles a player is holding and they are ready to respond to messages. The current role is the role that is directly responding to messages. That is to say, a role player can hold many active roles at the same time and it only holds one current role at a time. By holding only one current role, we can avoid role-role conflicts [13]. 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.

### 5.3 Past roles

To express an agent's dynamic and evolving aspects, we cannot avoid time. To completely express a live agent, we need to express the agent's past, current, active roles and future roles.

To track an agent's status, a history record is required. A history record helps answer such a question: 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 the roles played by an agent in the past and this role has already been discarded before. Fig. 5 shows the roles and the time segment for an agent to play roles.

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 is up to the actual requirement. 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.

The horizontal axis stands for time and the vertical axis for roles. To view an agent's evolution with the above concepts, we can concentrate on the role transition of the agent along the time axis horizontally. In Fig. 4, 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. 5 were  $R_1$ - $R_3$ , the active ones are  $R_1$ ,  $R_4$  and  $R_n$ , and current one is  $R_n$ .

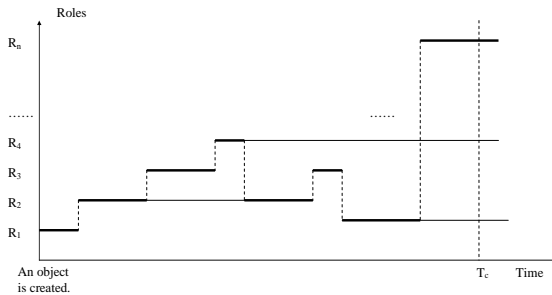


Fig. 5. The role play graph showing the roles an agent plays.

## 6. Challenges to Role-Based Intelligent Agent Systems

From the above discussed role-based architecture, we find that system design and system implementation can be separated clearly. This separation can help people specialize in easy-controlled knowledge and skill training. We may have professions in the near future: role-based architecture design, role-player design, and role-agent matching. To achieve this, we need to solve the fundamental challenges as follows:

- Role identification: find general roles and special roles for special research and application domains.
- Role specification: specify the incoming messages and outgoing messages for each role identified from the step before.
- Role relationship specification: the major relationships are the request/service, role ladders, and role conflicts. That is, some roles request and some roles serve; some roles are preliminary of some other roles; some roles cannot exist with some other roles.

We also need to answer the questions as follows:

- Should an agent be developed before roles are assigned or after roles are assigned?

In agent systems, agents are autonomous entities and adopted to the system automatically. Therefore, agents should collect abilities to play roles based on normal ideas in social lives.

- How could we design agents based roles before roles are assigned?

This is a question following the first question. Based on our definition of roles, we know that if an agent is assigned a role, it will possess both responsibilities and rights, some rights are required to take some responsibilities. Therefore, an agent can be designed with some assumptions of rights when taking responsibilities. This leads to a new design question.

- Should a role be assumed when an agent is designed?

The answer should be yes. This is a practical way to design agents before roles are assigned to the agents. This also demonstrates the importance of role negotiation, i.e., in the system integration, there is a need to negotiate roles between role designers and agent designers.

- Should an agent be developed with the discerning ability to know its goals and know how to reach the goals?

To answer this question, we need to know the system's requirement. As an intelligent system, agents should be autonomous and they should be able to adapt to the environment and cooperate with other agents to make the system evolve. Therefore, the agent designers should design agents by setting future roles as the goal for the agents, setting the qualifications to be promoted to a definite role.

A general process in an agent A could be as follows:

```
R = the initial role;
do {
  do {
    Play R;
  } while (A.credits < R.next().required credits);
  R = R.next();
} while (R!=finalRole);
```

From the above discussion, we find that agent design is simplified by introducing roles and role structures. Conventional technologies and methodologies try to design and create very intelligent agents to complete complex intelligent tasks. In role-based agent systems, agents become entities that are just required to accomplish some simple tasks confined by roles. In this way, the tedious tasks to build complex intelligent agents are simplified.

## 7. Conclusions and Future Work

Roles are fundamental for agent systems or distributed intelligent systems. By role-based architecture, we know that a distributed system can be designed by role and role relationship specification. In a distributed computer environment, we can consider a

computer as an agent in the networked systems. Every computer added to the network is to play a role or a group of roles in order to make the distributed system perform better.

Role-based architecture is a great guideline for distributed intelligent systems builders and designers to plan and architect a distributed intelligent system.

Our E-CARGO model have 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. Our future work will be on a role-based robot soccer team. This should be a good way to exhibit and convince the usefulness and efficiency of the role mechanisms we have developed.

### Acknowledgements

This research is supported by the IBM Eclipse Innovation Grant Funding. Thanks to Bart Verzijlberg for his proofreading this article.

### References

- [1] Bowling, M. and Veloso, M., "Multiagent Learning Using a Variable Learning Rate", *Artificial Intelligence*, vol. 136, no. 2, Apr. 2002, pp. 215-250.
- [2] Castelfranchi, C., "Modeling Social Action for AI Agents", *Artificial Intelligence*, vol. 103, 1998, pp. 157-182.
- [3] Coulouris, G., Dollimore, J., and Kindberg, T., *Distributed Systems: Concepts and Design (4<sup>th</sup> ed.)*, Addison-Wesley, 2005.
- [4] Cristiano, C., "Modelling social action for AI agents", *Artificial Intelligence*. Vol. 103, no. 1-2, Aug. 1998, pp. 157-182.
- [5] Cyert, R. M and MacCrimmon, K. R. Organizations, Lindzey, G. and Aronson, E. eds., *The Handbook of Social Psychology, Vol. 1*, Addison-Wesley, 1968, pp. 568-611.
- [6] Gruver, W., "Technologies and Applications of Distributed Intelligent Systems", *IEEE MTT-Chapter Presentation*, Waterloo, Canada, 2004.
- [7] Hayes-Roth, B., "An Architecture for Adaptive Intelligent Systems", *Artificial Intelligence*, vol. 72, no. 1-2, Jan. 1995, pp. 329 – 365.
- [8] Heylighen, F., "Collective Intelligence and its Implementation on the Web: Algorithms to Develop a Collective Mental Map", *Computational & Mathematical Organization Theory*, vol. 5, no. 3, October 1999, pp. 253 – 280.
- [9] Lavender, R. G. and Schmidt, D. C., "Active Object: an Object Behavioral Pattern for Concurrent Programming," in *Pattern Languages of Program Design* (J. O. Coplien, J. Vlissides, and N. Kerth, eds.), Reading, MA: Addison-Wesley, 1996.
- [10] Louden, K.C., *Programming Languages: Principles and Practice (2<sup>nd</sup> ed.)*, Brooks/Cole, 2003.
- [11] Maslow, A., *Motivation and Personality (2<sup>nd</sup> ed.)*, Harper & Row, 1970.
- [12] Maes, P., "Artificial Life Meets Entertainment: Lifelike Autonomous Agents", *Communications of the ACM*, vol. 38, no. 11, Nov. 1995, pp. 108-114
- [13] Nyanchama, M. and Osborn, S., "The Role Graph Model and Conflict of Interest", *ACM Transactions on Information and System Security*, vol. 2, no. 1, 1999, pp. 3-33.
- [14] Papazoglou, M. P., "Agent-Oriented Technology in Support of E-Business", *Communication of the ACM*, vol. 44, no. 4, April 2001, pp. 71-77.
- [15] Webber, A.B., *Modern Programming Languages*, Franklin, Beedle & Associates, 2003.
- [16] Wooldridge, M. and Jennings, N. "Intelligent Agents: Theory and Practice", *The Knowledge Engineering Review*, vol. 10, no. 2, 1995, pp. 115-152.
- [17] Zhu, H. "Encourage Contributions by Roles", *IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2005, Hawaii, USA, pp. 1574-1579.
- [18] Zhu, H. "Role Mechanisms in Collaborative Systems", *International Journal of Production Research*, vol. 41, no. 1, Jan. 2006, 181-193.
- [19] Zhu, H. and Zhou, M.C., "Role-Based Collaboration and its Kernel Mechanisms", *IEEE Trans. on Systems, Man and Cybernetics, Part C*, to appear, 2006.