

A Role Agent Model for Collaborative Systems

Haibin Zhu

Member, IEEE, Dept. of Computer Science, Nipissing University

100 College Dr., North Bay, ON P1B 8L7, Canada

haibinz@nipissingu.ca

Abstract

Collaboration based on computers involves two factors, human users and computer systems. When we hope to build a feasible and efficient collaborative system, we must consider both factors. Currently, it has increasingly become apparent that the collaborative system without human involvement is not only unfeasible, but also undesirable from points of view of reliability and safety. Thus, separating agent from object has been drawing attention to design collaborative systems in recent years.

In this paper, we propose a new role agent model that combines object, agent, and role concepts. It defines roles clearly and formally, considers more about human users' participation in collaboration with roles, and provides facilities to apply roles with role agents. With exact roles defined and the help of role agents, a human user may easily contact a collaborative system, clearly understand how to use the system, and obtain higher productivity of collaboration.

Keywords: *Role, Role Agent, Collaborative Systems.*

1. Introduction

CSCW (Computer-Supported Collaborative Work) applications or collaborative systems involve both computers and people. It has increasingly become apparent that the collaborative system without human involvement is not only unfeasible, but also undesirable from points of view of reliability and safety. Thus, separating agent from object has been drawing attention to design collaborative systems in recent years [6].

In the CSCW community there is no doubt about the importance of roles [2]. However, not much research about the role concept has been performed [20]. Only some authors tried to use roles in workflow control to improve coordination [3, 4, 5].

In the object community, there is widespread agreement that the concept of role is important for object modeling [11]. However, because there is no fundamental

discussion and well-defined role concept, traditional OO (Object-Oriented) models have difficulty expressing roles [10, 19].

The object-oriented role model identifies archetypal structure of elements and describes it as a corresponding and reoccurring structure of role. It hopes to express easily collaboration among objects and takes the natural meaning of roles [18].

However, a collaborative system will execute among objects and people. In a CSCW system, there are always many objects that represent the states of collaborators involved. These objects are different from other objects, such as documents and other different resources.

As a matter of fact, role definition is itself a difficult job in managerial and behavioral sciences and many consultant companies offer services such as position or role descriptions to help enterprises find a correct person to do a special job [1, 12, 13].

In RBAC (Role-Based Access Control), a role is defined as a semantic construct forming the basis of access control policy [17]. However, we still need to define and clarify the role concept in collaboration with our specific requirements.

Because agents are identified with autonomous active objects and incorporate properties like concurrency, reactivity, and autonomy [8], we believe that agents can certainly help the interactions between human users and collaborative systems.

In this paper, we hope to propose a new model that combines objects, agents and roles to overcome the difficulties in the interactions between human users and traditional CSCW systems. Our key point is to view a role as an independent concept in collaborative systems.

The basic idea of our approach is that if users log into a collaborative system that can designate clearly what objects they can access with specific rights, and can designate which users they can manage or communicate with, they can accomplish their jobs meaningfully and efficiently. In this style, collaboration is done successfully. In our approach, a human user is represented as one agent in a system.

The following scenario is one of our model's typical applications.

- A collaborative system built with this model is installed on a server.
- Some collaborators create objects and define roles in the system.
- Each collaborator involved in collaboration works on his/her client computer with logging into the system.
- The collaborator will directly use the interface provided by a role agent.
- Through the role agent that may be tuned by other collaborators (with the help of other role agents), the collaborator will access objects, contact other role agents, and contribute to the collaboration.
- The result of the collaboration is reflected by the states of objects in the system.

In the second section, we discuss the key concepts role and role agent in our model. Then, we introduce other basic concepts in a collaborative system with role agents. After we define the basic components in our role agent model abstractly, we discuss the interface design of some primitive roles in collaborative systems. Before the conclusion, we mention some related research on roles and agents. Finally, we conclude that the role agent model can bring us more advances in building collaborative systems.

2. The Characteristics of Roles and Role Agents

In behavioral science, a role is defined as a prescribed pattern of behavior expected of a person in a given situation by virtue of the person's position in that situation [12].

Within an organization individuals fill a specific position. A position in this respect represents a specific "seat" which has certain privileges and accompanying responsibilities [5]. A role should be defined as a set of capabilities and an expected behavior [6].

Nevertheless, it is difficult to describe a role clearly and strictly because natural languages are by their very nature ambiguous. For this reason, different persons in identical positions may make different contributions.

In an information system, we may clearly specify the responsibilities and capabilities of a person with some special technical methods. This possibility is due to the exactness of computer languages and the limited computer resources a person can control and access at any time.

A role can be defined 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 [2].

A role can also be defined as an abstraction of the behavior of an object, which consists of a subset of the interactions of that object together with a set of constraints on when they may occur [10].

In a RBAC system, a role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role [9].

In UML, a role is defined as a named specific behavior of an entity participating in a particular context [18].

In OO models, a role type describes the view one object holds of another object. One can say that the object plays a role specified by a role type [16].

The above definitions have difficulties in describing clearly the responsibilities a human user should take in an organization or in collaboration.

In the collaboration view, a role is the part of an object that fulfills its responsibilities in the collaboration. A role always belongs to a specific larger behavior that involves other roles, called a collaborative behavior [10].

In fact, in a collaborative system, a role is an independently existing entity or object. The relationship between a role and a human user is similar to that between a costume and a human being. A role exists whenever there is a human user performing it, just as there is a costume whenever there is a human putting it on.

Based on this view of roles, we characterize a role as follows.

- A role is an independent object in a system. We can define it separately.
- A role can be performed by a human user or by many human users at the same time.
- A role can be created, changed and deleted by a human user with a special role.
- A role is mainly concerned with an interface between human users and systems.
- A role is enacted in groups.

With the role concept shown above, we can introduce a special agent as a service provider to a person with a relevant role. We call it a role agent. As special agents, role agents mainly support the functions such as reminding human users of responsibilities, restricting the accessibilities to objects, and transferring messages to other objects and role agents. They accomplish these tasks by their knowledge about the roles in the system. Using a similar analogy method as above, a role agent is similar to a special server who serves a human that wears a special costume!

Therefore, we can characterize a role agent as follows.

- A role agent is created when a human user logs into a system. That means a human user's logging in results in the creation of a role agent, which is actually a combination of a role and an agent.

- A role agent is destroyed when a human role transfers to another role or the human user logs out. The similar idea in normal life is that, when a human changes into another costume, a new server will serve him or her.
- For each human user who has registered, there is an agent for him or her. There is one role agent for each logged human user. However, there is no role agent for a human user who has not logged into the system.
- In contrast to a role, a role agent is temporal and dependent on an agent. A human user may transform to other roles based on his/her interaction with the system.
- A role agent may provide services or respond to human user's messages autonomously.

3. Basic Concepts in the Role Agent Model

By introducing roles and role agents, our model hopes to propose a method to define roles clearly and exactly at first, then apply the roles in collaboration. Instead of taking roles and objects with the same identities [16], we consider roles as representative tags for human users to access system resources.

In collaboration, there are human factors and system factors. We call the persons who are involved in the collaboration *human users*. We call the system the human users are using *a collaborative system*.

Even though a human user in collaboration cannot be changed physically, his or her role may be changed in collaboration. Therefore, in a collaborative system, we can define *a role* as a definite state of a human user.

Roles are generally defined by responsibilities. From an object-oriented viewpoint, we can use *messages* a human user can send to express a role's responsibilities. Therefore, a role is expressed by *an interface* between a human user and the system. The interface is actually composed of a list of messages to *objects* in the system. The interface will change based on the messages a human user has issued. There is a special *agent* that represents the existence of a human user in the system. A role agent is an agent holding a role and a service provider to a human user. We introduce a *group* as a set of agents that facilitate a role's specification, because roles are generally defined and enacted within groups [13].

The shared objects are changed by the messages issued by the human users, and the human users finally produce a result of collaboration by these objects.

In an object-oriented view, a collaborative system is itself an object that is composed of objects. Every object has a list of messages the human users or other objects can send to it. An object provides a different list of messages to different human users based on their roles.

Therefore, a collaborative system provides different interfaces to a human user when he/she changes his/her role.

In summary, in our *role agent model for collaborative systems*, there are concepts as follows.

- An *object* can be used to express everything in a collaborative system [14].
- We accept *classes* as templates of objects [14].
- A *human user* is a person who is involved in collaboration.
- A *message* is a command that may be issued by a human user.
- An *interface* is a list of messages sent by a human user to objects in the system or to the system itself.
- An *agent* is a special object that represents a human user.
- A *group* is a set of agents.
- A *role* is an entity that expresses the responsibilities and capabilities a human user holds. It is expressed by a specific interface including messages to classes, objects and groups.
- A *role agent* is a special agent relevant to a specific role.

With the introduction of roles, a collaborative system will support the collaboration mode depicted in Figure 1, where each human user should log into the system with a role. With a role, he/she uses a specific interface provided by a role agent to access objects in the system, to send messages and transfer to other roles. With the interactions of between the human users and the system, collaboration takes place and produces the outcomes expressed by the objects in the system.

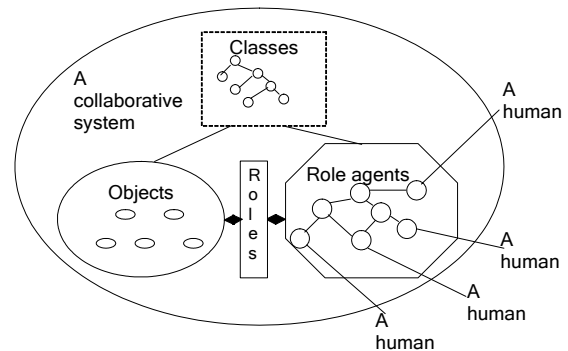


Figure 1. The collaborative mode based on roles

In agent systems, an agent will do a job according to its knowledge to the whole system. In a collaborative system, a human user will ordinarily accomplish his/her tasks by his/her own decision. In our model, therefore, role agents are helpers of human users and provide special interfaces between human users and the system.

In the next section, we hope to describe every concept with a formal definition, which can form components of a collaborative system built with this model.

4. Definition of Basic Components of the Role Agent Model

From the above discussion, we can model a *collaborative system* as a tuple Σ .

$\Sigma ::= \langle \{C\}, \{O\}, \{A\}, \{R\}, \{A_r\}, \{G\}, s_0, \{H\} \rangle$, wherein,

- $\{C\}$ is a set of classes;
- $\{O\}$ is a set of objects;
- $\{A\}$ is a set of agents;
- $\{R\}$ is a set of roles;
- $\{A_r\}$ is a set of role agents;
- $\{G\}$ is a set of groups;
- s_0 is the initial state of a collaborative system; and
- $\{H\}$ is a group of human users.

From the view that everything in the world is an *object* and every object has a *class* [14], an object [15] must be

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

and 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 is not controlled directly by people; and
- “collaborative”, i.e., collaborative relationships between objects arise.

In daily life, “communicative” might be implicit. For example, a cart can be moved only when one pushes it. This situation can be seen in the following way. It is the cart that accepts a message “go” from someone, and it goes by responding to this message. Hence, the “communicative” property of a cart is implicit and “pushing” is a message passing or a communicating style. That’s why we use agents to express the objects that possess all the properties discussed above [2].

Considering the general meaning and properties of objects, we can express a *class* by a quadruple. We can define a *class* expressed by C in a quadruple.

$C ::= \langle \mathcal{N}, \mathcal{D}, \mathcal{F}, \mathcal{X} \rangle$, where

- \mathcal{N} is the identification of the class;
- \mathcal{D} is a space 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 composed of all messages to the objects of this class.

In the following discussion we emphasize that an identification has two parts, one is an internal identification, and the other is an external one generally a character string.

With introducing the concepts of inheritance, we can redefine a *subclass* C_s ($\{C_s\}$ is a subset of $\{C\}$) as follows.

$C_s ::= \langle \mathcal{N}, \{C\}, \mathcal{D}, \mathcal{F}, \mathcal{X} \rangle$, where

- $\{C\}$ denotes a set of superclasses identified by their identifications or names, and
- $\mathcal{N}, \mathcal{D}, \mathcal{F}$, and \mathcal{X} have the same meanings as those of C .

With the class defined, we can define an *object* based on a class. An *object* is defined as $O ::= \langle \mathcal{N}, C, S \rangle$, wherein,

- \mathcal{N} is the identification of an object;
- C is the object’s class identified by the class identification or name; and
- S is a body or an actual space whose values are called attributes, properties, or a state.

An agent is defined as an entity consisting of a set of provided services [2].

In our model, an *agent* \mathcal{A} is a special object that represents a human user that is involved in collaboration and may log into the system.

$\mathcal{A} ::= \langle \mathcal{N}, C_a, S \rangle$, wherein,

- C_a is a special class that describe the common properties of human users; and
- \mathcal{N} and S have the same meanings as that in the definition of O .

The C_a of an agent will reveal the differences from ordinary objects. We can also introduce special identifications to distinguish agents from objects.

To distinguish agents from objects, we can use message responses. Objects respond to messages by initiating their class methods, but agents may respond to messages by showing the messages to the human user it represents, or just transfer these messages to other agents. To support the functionalities of role agents, an agent class should have the knowledge about groups, classes and objects in of the system.

To facilitate interactions among objects, we use messages. A *message* \mathcal{M} can be defined as $\mathcal{M} ::= \langle \mathcal{N}, \mathcal{T}, \{O\} \rangle$, wherein,

- \mathcal{N} is the identification of the message;
- \mathcal{T} means the receiver of the message expressed by an identification of a class, an object or a group; and
- $\{O\}$ is a set of objects taken as parameters.

In a traditional OO model, we concentrate mainly on objects and their classes, because executable programs will run automatically with little interaction with human users. Traditional OO paradigm emphasizes the messages accepted by a class of objects. However, it did not consider much about the messages an object may send

out. In our role agent model, we emphasize that a role is mainly concerned with the messages that may be sent out.

Agents taking part in a cooperation process play at least one role. A role requires the agent to have permissions and capabilities, whereas an agent playing a role commits itself to the obligations specified by the role. During the execution of a cooperation process an agent can change its role. This allows us to have roles designed for a particular purpose and therefore easy to maintain [2].

A *role* can show the specialties of some human users, it provides human users messages to access objects, classes and groups. A role \mathcal{R} can be defined as $\mathcal{R} ::= \langle \mathcal{N}, \mathcal{P}, I \rangle$, wherein,

- \mathcal{N} is the identification of the role;
- \mathcal{P} is a description in a natural language for the role;
- $I ::= \langle \mathcal{M}_o, \mathcal{M}_c, \mathcal{M}_g \rangle$ denotes a set of messages, wherein, \mathcal{M}_o , \mathcal{M}_c , and \mathcal{M}_g express different sets of out messages for this role, that is, messages to an object, messages to a class, and messages to a group. We can extract these messages from the interfaces of classes. In implementation, it is a list of links or buttons that a human user can click; and
- \mathcal{M}_o , \mathcal{M}_c , and $\mathcal{M}_g ::= \{M\}$ that means a set of messages. In \mathcal{M}_c and \mathcal{M}_g , we can divide the messages into two categories. One is *any-messages*, and the other is *all-messages*. By any-message we mean that the message may be sent to any member of the group or any instance of the class; and by all-message, we mean that the messages should be sent to all the members of a group or all the instances of the class.

The difference between the I of \mathcal{R} and the X of C is:

- The \mathcal{N}_s of messages in the X of a class C are the same. In other words, the messages will be sent to an instance of this class; but
- The \mathcal{N}_s of messages in the I of a role \mathcal{R} are different, they may be an object, a group, or a class.

A *role agent* is an agent with a specific role attached. It can accept messages from and can send messages to other agents. An agent in a collaborative system is a representative for a human user, and it has all the information the human user hopes to store in a system.

One important aspect for a role agent is that it may not make a decision for the human user it represents, and it may show many possible choices for the human user and let him/her to make the decision. From the computer interface's viewpoint, a role agent should show a list of menus or buttons for the human user to click.

Another important function of a role agent is that it must help a human user to participate in collaboration. For example, if a human user wants to send a message to all members of a group, it should find all the members and send out the message to them; if a human user wants

to send a message to any instance of a class, it should find an instance of a class to send the message and to make sure that the message touches the instance.

A *role agent* \mathcal{A}_r can be defined as $\mathcal{A}_r ::= \langle \mathcal{N}, \mathcal{R}, \mathcal{A} \rangle$, wherein,

- \mathcal{N} is the identification of the role agent;
- \mathcal{R} is the human user's role identified by the role's identification or name; and
- \mathcal{A} is an agent that represents a human user.

In reality, human users work in a group and hold roles. In our model, we define a group a set of agents. $G = \langle \mathcal{N}, \{A\} \rangle$, wherein,

- \mathcal{N} is the identification; and
- $\{A\}$ is a set of agent identifications.

We call an agent that belongs to a group a *member*.

The *initial state* s_0 is expressed by initial values of all the components, such as, primitive roles, initial objects, and so on.

With the participation of *human users* $\{H\}$, a collaborative system Σ evolves, develops and functions.

5. The Interface of a Role Agent

From the above, a role is an internal structure, and a role agent provides a human user a special interface corresponding to the role the human user is performing. In the basic structure of collaboration based on role agents, we can support interactions between human users and role agents by defining a specific interface.

The interface of a general role should support the interfaces as follows. That is to say, a role agent will show this kind of interfaces to a human user with a definite role.

- The human user's current role.
- The role's description.
- The human user's profile such as name, title, etc.
- The list of classes, groups and objects the human user can access.
- The message sending facilities:
 - The all-messages to classes and groups
 - The any-messages to classes and groups
 - The messages to objects

This interface seems similar to an email tool interface. However, this interface supports broader categories of messages including ordinary text messages. For example, in this interface a human user can send a message to all instances of a class, and this message will lead to the changes of the states of all the instances of the class. The messages also include those which can help human users get the states or contents of selected objects.

Please note that, without roles and role agents, it would be very difficult to let human users to send out

messages to any instance or all instances of a class, or to any member or all members of a group.

6. Primitive Roles to Support Role Agents

In a collaborative system, to support the role agents, we must design some predefined roles (also called primitive roles), such as role definition role, group definition role, class definition role, and object instantiation role.

The interface of a role definition role should include the following:

- The role's name.
- The role's description.
- The list of classes, groups, and objects the human user can access.
- For a class, list the messages
 - For each message, specify it to all instances or to any instance.
- For a group, list the messages
 - For each message, specify it to all members or to any member.
- For an object, list the messages.

The interface of a class definition role should include the following:

- The class's name.
- The class's description.
- A list of classes to be selected as superclasses of this class.
- A form to specify the data structures, such as, names and their types or classes
- A form to specify the functions, such as, function names, return types, and parameter names and parameter types. We can describe the processing of relevant functions by applying specific programming languages, such as, C++ or Java.

The interface of a group definition role should include the following:

- The group's name.
- The group's description.
- The list of agents the human user can select as members of this group.

The interface of an object definition role should include the following:

- The object's name.
- The object's description.
- The list of classes the human user can select as a class of this instance (object).
- After a class is selected, list all the instance variables that the human user can specify their values.

7. Related Works

Cesta and D'Aloisi conceived that the design of interfaces presents a high degree of complexity since the relationship with the user and his/her needs is very critical. They proposed to build interfaces as personal agents [7].

Cabri, Leonardi and Zambonelli defined a role with four elements in their XRole language [6]:

- Name. The name given to the role.
- Description. A high-level description of the role.
- Keyword. Zero or more keywords can be used to identify the role.
- Action. It describes actions available to the agent if it assumes this role.

The above description is coming from the intuitive view to a role in managerial and social activities. In computer-supported collaboration, the actions taken by a role is more important than other attributes.

In XRole, the actions are described with a list of action names and comments. However, the roles do not connect directly to objects human users hope to control and manage. This description is too general to really support the jobs of a human user. The action is only a textual description.

Genilloud and Wegmann discussed role as a modeling concept completely in an object-oriented view [10].

Depke, Heckel and Kuster provided a method to improve the agent-oriented modeling process by roles. They introduced a concept of roles in order to support the transition in a systematic way and thereby improve the agent-oriented modeling process [8].

Yu and Schmid proposed a conceptual framework for agent oriented and role based workflow modeling. In their paper, they view a business process as a collection of autonomous, problem solving agents which interact with others when they have interdependencies [19].

The above contributions motivated us to construct our role agent model.

8. Conclusion

Providing *role agents* can help human users to know exactly what they can do in a computer-supported collaboration. With the help of role agents, different human users can get specific supports for his or her current positions. The *role agent model* considers both the computer and human factors, and it defines human users as a part of the system. More important is that, with the *role definition role agent*, a human user can define new *roles* in the system and make the collaboration more practical and applicable. Even better is that, with roles and role agents, human users can easily send out messages to groups and classes.

In this paper, we discussed the basic concepts of roles in a collaborative system, specified the basic components of it, and emphasized the importance of role agents. Our key point is to view a role as an independent entity that can be performed by human users.

In the role agent model, we define roles clearly and formally, consider more about human users' participation in collaboration with roles and provide facilities to apply roles with role agents. With exactly defined roles and the helps of role agents, human users may easily contact a collaborative system, clearly understand how to use the system, and obtain higher productivity of collaboration.

This paper also presented an implementation guide by defining some primitive roles.

9. Acknowledgement

This research is supported by the Research Funding of Nipissing University (No.: 10-3287-42195).

10. References

- [1] Biddle, B.J., and Thomas, E.J.(Ed), *Role Theory: Concepts and Research*, John Willey & Sons, Inc., 1966
- [2] Becht, M., Gurzki, T., Klarmann, J., Muscholl, M., "ROPE: Role Oriented Programming Environment for Multiagent Systems", *Fourth IECIS International Conference on Cooperative Information Systems*, Sept. 1999, pp. 325-323
- [3] Bertino, E., Ferrari, E., and Atluri, V., "A flexible model supporting the specification and enforcement of role-based authorization in workflow management systems", *Proceedings of the second ACM workshop on Role-Based Access Control*, Fairfax, Virginia, USA, 1997, pp. 1-12
- [4] Bertino, E., Ferrari, E., and Atluri, V., "The specification and enforcement of authorization constraints in workflow management systems", *ACM Transactions on Information and System Security*, vol. 2, no. 1, Feb. 1999, pp. 65-104
- [5] Botha, R. A., and Eloff, J. H. "Designing Role Hierarchies for Access Control in Workflow Systems", *25th Annual International Computer Software and Applications Conference (COMPSAC'01)*, Oct. 2001, pp. 117-122
- [6] Cabri, G., Leibardi, L. and Zambonelli, F., "Implementing Role-based Interactions for Internet Agents", *3rd IEEE Symposium on Applications and the Internet (SAINT 2003)*, Melbourne (FL), IEEE CS Press, Feb. 2003
- [7] Cesta A. and D'Aloisi, D., "Building Interfaces as Personal Agents", *SIGCHI Bulletin*, vol. 28, no. 3, July 1996, pp. 108-113
- [8] Depke, R., Heckel, R. and Kuster, J. M., "Improving the Agent-Oriented Modeling Process by roles", *The Fifth International Conference on Autonomous Agents*, May 2001, Montreal, Quebec, Canada, pp. 640-647
- [9] Ferraiolo, D. F., Barkley, J. F., and Kuhn, D. R., "Proposed NIST standard for role-based access control", *ACM Transactions on Information and System Security (TISSEC)*, Vol. 4, No. 3, August 2001, pp. 224-274
- [10] Genilloud, G., and Wegmann, A., "A Foundation for the Concept of Role in Object Modeling", *Proceedings of the 4th International Enterprise Distributed Object Computing Conference(EDOC2000)*, Makuhari, Japan, September, 2000, pp. 76-85
- [11] Gottlob, G., Schref, M, and Röck, B., "Extending object-oriented systems with roles", *ACM Transactions on Information Systems (TOIS)*, vol. 14, no. 3, July 1996, pp. 268-296
- [12] Hawkins, D. I., Best, R.J., and Coney, K. A., *Consumer Behavior, Business Publications, Inc.*, Plano, Texas, 1983
- [13] Hellriegel, D., Slocum, J.W., Jr., and Woodman, R. W., *Organizational Behavior*, West Publishing Co. St. Paul, Minnesota, 1983
- [14] Kay, A. C. , "The early history of Smalltalk", *The Second ACM SIGPLAN History of Programming Languages Conference, ACM SIGPLAN Notice*, vol. 28, no. 3, March 1993, pp. 69-75
- [15] Meyer, B., *Object-Oriented Software Construction*, Prentice Hall, 1988
- [16] Riehle, D. and Gross, T., "Role Model Based Framework Design and Integration", *Proceedings of OOPSLA'98*, Vancouver, Canada, 1998, pp. 117-133
- [17] Sandhu, R., and Munawer, Q., "The ARBAC97 model for administration of roles", *ACM Transactions on Information and System Security*, vol. 2 no. 1, 1999, pp. 105-135
- [18] Wegmann, A. and Genilloud, G., "The role of 'Role' in Use Case Diagrams", *Proceedings of Third International Conference on The Unified Modeling Language. Advancing the Standard(UML 2000)*, York, UK, October 2000, pp. 210-224
- [19] Yu, L. and Schmid, B. F., "A Conceptual Framework for Agent Oriented and Role Based Workflow Modeling", *NetAcademy*, <http://www.netacademy.org>
- [20] Zhu, H., "Some Issues of Role-Based Collaboration", *2003 Canadian Conference on Electrical and Computer Engineering (CCECE'03)*, Montreal, Canada, May, 2003