

Fundamental Relations among Roles and Agents in Role-Based Collaboration

Haibin Zhu

*Department of Computer Science and Mathematics, Nipissing University
100 College Drive, North Bay, Ontario, P1B 8L7, Canada*

E-mail: haibinz@nipissingu.ca

Abstract

A society requires a government to manage, supervise and guide the activities of its members. Every system requires controls and supervision. A collaborative system also requires management and control. Role-based systems have the same requirements.

In role-based collaboration (RBC), roles are the major media for interaction, coordination, and collaboration. A center is required to deal with the management of roles and role players, and the control of messages sent from roles to roles. This center is called as a role engine. In this paper, the basic functions of a role engine and the fundamental relations in RBC are discussed. These relations must be clearly specified before building a role engine and a role-based system.

Keywords: roles, role engine, agents, relations, and role-based collaboration.

1. INTRODUCTION

Role-Based Collaboration (RBC) is a computational thinking methodology that mainly uses roles as underlying mechanisms to facilitate abstraction, classification, separation of concern, dynamics, and interactions. It will find wide applications in different fields, such as, organizations, management systems, systems engineering, and industrial engineering. It encompasses computational concepts, methods, models, algorithms, and tools. Based on roles, RBC is such an emerging methodology to facilitate an organizational structure, provide orderly system behavior, and consolidate system security for both human and non-human entities that collaborate and coordinate their activities with or within systems. RBC is a way to understand the complexity in natural, built, and social systems, because RBC enables the design, synthesis, and control of novel complex engineered systems.

A country requires a government to manage, supervise and guide its people. A society requires a board of governors to regulate the recruiting, the behaviors and the collaborations among its members. Every system requires controls and supervision. A collaborative system also requires management and control. A center to manage

and control is required in role-based systems based on the same requirement.

An organization can be expressed in terms of a set of roles that determine the social position of agents within such an organization and their relations to other members of the organization. Agents should be designed in a way that enables them to reason about the social position they gain by adopting a specific role, and what the correct behavior in such a role would be [5].

In social life, roles are abstract and implicit. It is people but not roles that interact with each other in collaboration. People transfer roles when they are collaborating. This role transfer is implicit and there is no evidences for others to view easily and clearly if the people do not declare. In some cases, even people declare they play roles, others may not confidently know what role the people are playing. Some external objects such as chairs, tables, rooms, and titles are very weak flags to express the roles. People may be in one place for one role and do something of other roles. This demonstrates the requirement of a role specification methodologies and tools. In role-based collaboration, roles are the major media for interaction, coordination, and collaboration. A role engine is required to deal with the management and control of messages sent from roles to roles. In computer-based systems, roles can be made into concrete facilities and there are clear and explicit flags for other agents to know what roles the agents are playing so as to collaborate efficiently.

It is required to specify the roles and role relations before designing agents and system integration. However, how to specify roles and role relations is a challenge that has not been completely investigated and it becomes a difficult problem that must be solved to make these software development methodologies to advance.

Roles have three functions: specify special behavior (as interface roles), form the behavior of an agent (as process roles), and set a place for an agent in a group (or define the inter-relations among agents). Although many mention the relations among roles, there is no complete and comprehensive discussion on role relations.

Role specification includes two aspects: one is the content of a role and the other one is its relations with

other roles. Role relation specification significantly affects almost all the aspects of role-based systems, such as, *role modification, role assignment, role transfer, role execution and role interactions*. To clearly specify role relations is fundamental to system analysis, system design and system construction.

In this paper, the basic functions of a role engine and fundamental relations in the design of a role engine to support RBC are discussed. With a role engine, role-based collaboration among agents would become true. Our assumption is that an agent has only one processor and can only do things sequentially, i.e., at one time, an agent can only play one role.

In the following discussions, agents and people are used interchangeably. If the collaboration is among agents, they are autonomous agents. If the collaboration is among people, agents are representatives of people in collaboration.

This paper is arranged as follows. Section 2 depicts the basic functions of a role engine; Section 3 restates and revises the definitions of roles and agents of our pre-proposed model E-CARGO; Section 4 demonstrates the relations among roles; Section 5 defines the relations between roles and agents; Section 6 specifies the relations among agents; Section 7 presents the properties of an RBC system; Section 8 discusses related work; and Section 9 concludes this paper and proposes the future work.

2. THE FUNDAMENTAL FUNCTIONS OF A ROLE ENGINE

A role engine can be understood in the same way as a Prolog inference machine. For example, to use a Prolog system, people only need to write rules and facts. The Prolog inference machine will search the result. Similarly, to implement role-based collaboration, based on the proposed role engine, people could only specify roles and create agents based on role specifications. When agents are put into the role engine, the engine will drive agents work properly to obtain their goals by collaborating with other agents.

A role engine should do the followings:

- Managing roles (create, delete, and modify);
- Managing agents (create, delete, and modify);
- Manage groups (create, delete, and modify);
- Assigning roles to agents;
- Dispatching messages to agents; and
- Checking the consistence in the system.

A role engine is a platform for agents to collaborate. On this platform, agents work for the system by playing roles. By the running of the role engine, all the agents are driven to contribute and work diligently for a system that is a solution for a real-life problem. A role engine should

possess role dynamics, facilitate role transfer, and support role assignments, interaction and presentation,.

2.1 Role Dynamics

Everything exists in the world for a special reason. Every activity in the world is instigated for some reason. These reasons form driving forces for new things to be created and for new actions to be taken. In the society, people in an organization with good dynamics will work actively and collaboratively toward the common goal of the organization. In contradiction, people in an organization without good dynamics cannot work as effectively and may not have a clear understanding of the goals necessary to make the organization competitive.

It is the same in a computational system. Here, if agents are taken as computational components, a computational system is a multi-agent system. Multi-agent systems should encourage diversity of behavior in a population of cooperating agents [14]. This is one kind of requirement for agents. There are definite reasons for the addition of new agents to the system. With the well-built dynamics, agents will automatically follow the regulations of the RBC system and collaborate with each other to approach the common goal of the system.

To build a multi-agent system, the following questions should be answered: “How are agents created?”, “Why are agents created?”, “Why are agents transferred on the networks?”, “How are agents made pro-active?”. “How could a system obtain the best performance?” All these questions concern about the fundamental mechanisms to build and operate a system, i.e., dynamics.

2.2 Role Transfer

The flexibility and survivability of a group (of people, systems, or system components) is largely dependent of its role assignments and role transferability [18]. Role transfer is a complex event that may involve many relevant roles and it is a highly intelligent job for managers to deal with. When an organization encounters a crisis situation, not every one can transfer his/her roles. When one person transfers to a new role, his/her original role needs to be assumed by another if the role is still required in the system. This change might initiate a series of role transfers. For example, in a battle field, if a high rank officer is shot, injured or dead, a similar or lower rank officer is needed to play his/her role. The role played by the similar or lower rank officer may need to be played by another, etc. A highly-available distributed computer system requires its sub-systems, computers, and components to transfer their functions (or roles) to recover from a faulty state to a workable state. It is needed to duplicate them (sub-systems, computers, and components) to guarantee the similar requirements for tolerating faults. That is to say, one specific role should be played by more than one person (a sub-system, a computer, a system

component). One person (a sub-system, a computer, a system component) should be able to play more than one role [1, 3, 8].

Although the more the backups the better the systems, it is definitely required to find a best solution (locally or generally), i.e., the “most economic”, the “easiest” and the “fastest” ways to arrange “enough” backups. To solve such an optimization problem, the following questions should be answered clearly: What does “critical roles” mean? What does “critical people” mean? How are “critical roles and critical people” expressed? How is a crisis specified? Who is the best person to play a critical role? What are the criteria to evaluate if a person is qualified to play a role? How many roles should a person play or potentially play to deal with a crisis? Which roles should a person play? How can a crisis avoided by appropriate role transfer?

Role transfer can also be applied in different aspects of highly available systems such as duplications, fault-tolerance, concurrency and parallelisms.

2.3 Role Assignment

In the business world, a person is required to have new knowledge, skills and habits to be qualified for a new position. Successfully finding a new job is dependent on similarities between new roles and those previously performed by a person [2], i.e., qualifications are the basic requirements for possible role-related activities.

Role assignment is the first event for RBC and is dependent on the qualifications of agents. There are two difficult aspects: when roles are defined as abstract interfaces, it is needed to match the abstract interfaces with the concrete things (syntactically and semantically) an agent can do; when roles are specified as concrete processes, it is needed to specify all the details of the roles exactly in both syntax and semantics and should be able to adapt to different agents of the system.

To solve the role assignment problem, it is needed to solve the problem of role definition and specification. Clear role definition and specification help a person collaborate by avoiding ambiguities and conflicts [3]. No body likes to work in a group without clear regulations and rules. A society with high efficiency and productivity should be well-organized, well-regulated and well-managed. Specifications are the key points. The first questions “how is a role defined?” and “how are roles specified?” should be answered.

After roles are defined well and specified well, role assignment can be solved based on these specifications and mining methodologies that extract formal specifications from files of natural languages related to the qualifications, responsibilities, and rights of agents. Role assignment can also be applied in knowledge extraction from vast amount of versatile raw data on the grid computing world.

In RBC, agent role matching is a difficult and fundamental problem. To assign an agent to a role, it is needed to evaluate if the agent is qualified to play the role. However, “qualification” is a complicated criterion that is composed many aspects. In the proposed role engine, a role specification methodology is taken as the solution to the qualification evaluation problem of agents. This methodology can also be used in service-oriented architecture to support service matching.

Any specification language or tool encounters the same problem at the beginning, that is, how to strictly express the semantics with simple syntax. For human users, simple syntax eases the using of the tool. In reality, ambiguous language problems occur everywhere at anytime. It requires strict semantics matching to check whether two objects are the same or not.

Roles are highly abstract concepts and they lack strict semantics definition, well-accepted connotation, and clear specification hitherto. This situation leads to the difficulty to apply role-based methodology into different areas. The basic methodology proposed in this paper is based on predefined roles, role specification based on the predefined role bases, and a role engine to evaluate agents’ qualification to play a role.

Because the assignment of roles to agents is dynamic in a multi-agent system, i.e., the roles can be re-assigned to other agents based on the changing of the environment. Therefore, assigning roles to agents also needs to keep the following properties:

- Fairness: the engine should assign roles to agents based on a fair rule. It should similarly avoid starvation or overloading, where starvation means that an agent has not received role assignments for a time longer than a limit while overloading means that an agent is assigned too many roles in a limited time.
- Balanced workload: the engine should assign roles to qualified agents according to the workload of agents. Special roles may also require many agents to play to save time.
- Least conflict: the engine should guarantee that the roles assigned to agents create the least opportunity for agents to have conflicts in sharing information when they are playing roles.

2.4 Role Interaction

With a role engine, interaction and collaboration are facilitated by roles. The role engine controls the messages exchanged among roles. Based on our E-CARGO model (Zhu & Zhou, 2006), interaction is implemented by issuing messages. Dispatching messages to agents is highly intelligent task to be accomplished by a role engine and its roles. It needs to consider several properties:

- Fairness: the engine should dispatch messages evenly to peer agents (see **Definition 29**). It

should avoid starvation or overloading, where starvation means that an agent has not received messages for a time longer than a limit while overloading means that an agent receives too many messages in a limited time.

- Consistency: the engine should check the consistency of role hierarchies. When a new role hierarchy is added, the system should be kept consistent (see **Definition 38**).
- Completeness: sometimes, the engine should dispatch a specific message to all the agents of the targeted group.

2.5 Role Presentation

Roles are finally presented to users. They should be easily understandable. The specification of roles should consider the easy implementation of role presentation. Role presentation should consider more on the aesthetics, intuitions, and the human factors.

Similar to other human user interface requirements, role presentation has the following requirements:

- Easy to understand;
- Easy to remember;
- Used to support personalized or customerized user interface; and
- Presented in a multi-media style, such as text, image, audio, video and animation presentations.

To meet the requirements as above, iconizing different roles are beneficial to role presentation. Some concrete roles can be easily expressed with icons such as a

police  , a waiter  , and a worker  .

However, it is difficult to express a computer professor, a software developer, and a system analyst. It is also very difficult to express a generalized concept “role” by an icon [16]. Even further, it is a very difficult task to design icons presenting as much information as that in a role specification. Therefore, tables, lists, and graphs are needed to present roles.

3. E-CARGO MODEL AND REVISIONS

Based on the E-CARGO model [17], software development is considered as role-based collaborative activities. In E-CARGO, a system Σ 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. In such a system, \mathcal{A} and \mathcal{H} , \mathcal{E} and \mathcal{G} are tightly-coupled sets. A human user and his/her agent play a role together. Every group should work in an environment. An environment regulates a group. With this

tight coupling, it is emphasized that a role-based collaborative system is composed of both computers and human beings.

From the viewpoint of role-based collaboration, software development is considered as role-based collaborative activities. Σ can be considered as a software development team. With the participation of people \mathcal{H} , such as joining in a team Σ , accessing objects of the team, sending messages through roles, forming a group in an environment, Σ evolves, develops and functions. The results of the team work are a new state of Σ that is expressed by the values of $C, O, \mathcal{A}, \mathcal{M}, \mathcal{E}, \mathcal{G}$, and \mathcal{H} , where, $\langle C, O, \mathcal{M}, \mathcal{R} \rangle$ forms the role-based software product.

In E-CARGO, agents are considered to have only one central processing unit and they can only process messages or play one role at a time.

In this paper, roles and agents are concentrated on. Roles are taken as abstract interface definition as proposed in the E-CARGO model. The process meanings of roles are not applied. Agents are role players. The definitions of an agent and role are revised as follows, where, if χ is a set, $|\chi|$ is its cardinality: $a.b$ means b of a or a 's b .

Definition 1: *role*. A role is defined as $r ::= \langle n, I, \mathcal{A}_c, \mathcal{A}_p, \mathcal{L}, \mathcal{N}_o, \mathcal{R}_r \rangle$ where,

- n is the identification of the role;
- $I ::= \langle \mathcal{M}_{in}, \mathcal{M}_{out} \rangle$ denotes a set of messages, where \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{A}_c is a set of agents who are currently playing this role;
- \mathcal{A}_p is a set of agents who are potential to play this role;
- \mathcal{A}_o is a set of agents who used to play this role;
- \mathcal{R}_i is a set of roles interrelated with it (see **Definition 16**); and
- \mathcal{N}_o is a set of objects that can be accessed by the agents playing this role.

Definition 2: *agent*. An agent is defined as $a ::= \langle n, c_a, s, r_c, \mathcal{R}_p, \mathcal{R}_o, \mathcal{N}_g \rangle$, where

- n is the identification of the agent;
- c_a is a special class that describes the common properties of users;
- s is the profile of the person whom the agent is representing;
- 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 ($r_c \notin a.\mathcal{R}_p$); and
- \mathcal{R}_o means a set of roles that the agent played before; and

- \mathcal{N}_g means a set of groups that the agent belongs to.

All the current role and the potential roles of agent a (i.e., $a.\mathcal{R}_p \cup \{a.r_c\}$) form its repository role set, denoted as \mathcal{R}_a .

With the E-CARGO model, a role engine can be designed to support RBC in the way stated by Shakespeare “All the world is a stage ($\mathcal{E}, \mathcal{C}, \mathcal{O}$), and all the men and women merely players (\mathcal{A}); they all have their exits and entrances (\mathcal{G}); and one man in his time plays many parts (\mathcal{R}). (As You Like It, Act II, Scene 7)”. In the following discussions, $r_0, r_1, r_2, \dots, r_i, \dots, r_j, \dots$ ($i, j=0, 1, \dots, i \neq j \rightarrow r_i \neq r_j$) are used to express elements in \mathcal{R} and a, b, c, \dots ($a \neq b, b \neq c, \dots$) in \mathcal{A} .

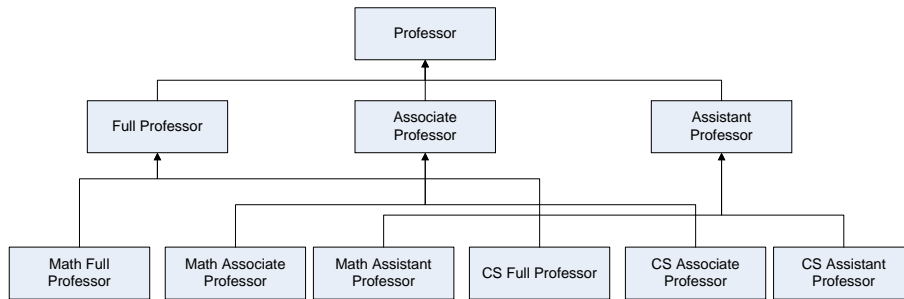
4. THE RELATIONS AMONG ROLES

The relations among people are complicated in social societies [6]. Role relations are more abstract relations than people’s relations. To classify and clarify role relations abstractly helps build an efficient role engine. Role relations are foundations to implement the functions of a role engine.

4.1 Role Classes and Instances

To specify role relations, it is required to specify what a role is. When roles are discussed, arguments about roles always occur. What are roles? Are roles classes? Are roles instances? What is a message to a role?

In role-based collaboration, we state that roles are instances of r in Definition 1 with properties of classes. r is actually a meta-class in object-orientation. On the aspect of an instance, a role has its concrete agents and messages. On the aspect of a class, its cardinalities ($|\mathcal{A}_c|, |\mathcal{A}_p|, |\mathcal{L}|$) are



Note: CS means Computer Science

Figure 1. Super roles and sub roles

In Figure 1, *Professor* is a super role of roles *Full Professor*, *Associate Professor*, *Assistant Professor* and *Math Professor*. *Professor* is a root role, and *Math Full Professor*, *Math Associate Professor*, ..., and *CS Assistant Professor* are leaf roles. In such a hierarchy, it is needed to clarify that an agent should be assigned with a leaf role, because leaf roles are concrete and executable by agents.

actually to express how many instances are created for agents to play it.

To classify, we define role class and role instances.

Definition 3: role class. Role classes are instances of r .

Definition 4: role instance. A role instance is an instance of a role class.

In the definition of role of the revised E-CARGO model, role r is in fact a meta-class for all role classes. When a new role is defined, it is actually an instance of r . When an agent plays a role, it plays a role instance linked with an instance of r . In RBC and E-CARGO, there is no concrete role instance. Role instances are actually implemented by agents playing a role class. The number of role instances is restricted by the cardinalities of a role. In the following discussions, the relations at the level of role classes are concentrated on.

4.2 Inheritance Relation

Roles themselves can be a classification tool. They are related in a classification hierarchy.

Definition 5: inheritance relation, super roles and sub roles. Role r_j is a super role of role r_i if r_i inherits r_j in the sense of object-oriented methodologies. Here, “inherits” means simple and whole inheritance. A *inheritance relation* denoted as Ω is a set of tuples of roles $\langle r_i, r_j \rangle$, where, r_j is a super role of r_i and r_i is a sub role of r_j .

Figure 1 shows the inheritance relations among a group of role classes.

Definition 6: root role and leaf role. Role r_i is called a root role if it has no super roles. Role r_i is called a leaf role if it has no sub roles.

It is noticed that this role relation is not a partial order [13], so are other role relations. The reflexive property is not held for this relation. For example, it is meaningless to say that a *professor* inherits from a *professor*.

Definition 7: message types. In RBC, a message can be three types: *any*, *all*, and *some*. Suppose r is a non leaf role, r_0, r_1, \dots , and r_{n-1} , are leaf roles whose super role is r . Any-messages to r should be sent to an agent

that plays a role of r_0, r_1, \dots , and r_{n-1} . *All-messages* should be sent to all the agents that play r_0, r_1, \dots , and r_{n-1} . *Some-messages* should be sent to some agents that play r_0, r_1, \dots , and r_{n-1} .

Definition 8: *successful message sending.* When a message is sent to a role, it is successful if the role covers the message pattern and it finally finds its receiver.

It is needed to check if this message is included in the role's service set or in the roles' sub role's service set. If the message is in the sets, the message sending is successful. In role-based systems, a role accepts and dispatch messages. A role cannot respond to messages directly.

4.3 Promotion relations

In a well-designed community, there are pre-designed role promotion relations. This relation encourages the members in the community to work hard to contribute. It means that an agent could play an upper level role only when it has played the lower level role before.

Definition 9: *role promotion, lower role and upper role.* Role r_j is an upper role of r_i if role r_i must be played before r_j is assigned. Vice versa, r_i is called a lower role of r_j . A *promotion relation* denoted as \mathcal{A} is a set of tuples of roles $\langle r_i, r_j \rangle$, where r_i is a lower role of r_j and r_j is an upper role of r_i .

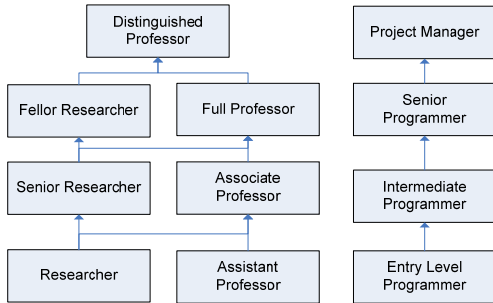


Figure 2. Examples of promotion relations

Role promotion is not natural but man-made. It is set in a role engine in order to be fair in role re-assignment of potential roles. It is set during the initialization and may be adjusted during collaboration.

Figure 2 gives two examples of promotion relations in the education and information technology fields.

4.4 Report-to Relations

A role provides many services. However, not all other roles can request its services and obtain instance responses. Some regulations are introduced to control the accessibilities of the services. For example, a team leader can tell a team member to complete a task but a team

member cannot tell a team leader what to do [4]. This leads to a report-to relation.

Definition 10: *report-to relation, supervisor role and supervisee role.* Role r_j is supervisor of role r_i if role r_i must respond to the requests from role r_j . Role r_i is a supervisee of role r_j . A *report-to relation* denoted as \mathcal{Z} is a set of tuples of roles $\langle r_i, r_j \rangle$, where, r_i is a supervisee role of r_j and r_j is a supervisor role of r_i .

A report-to relation is different from a promotion relation in that an agent plays a supervisee role may be impossible to be promoted to its supervisor role and a lower role may not be supervised by its upper role.

4.5 Request relations

Everybody serves others and everybody requests others' services. Therefore, an evident relation between roles is the request role.

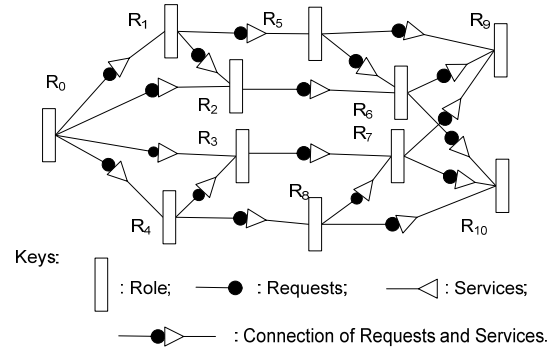


Figure 3. A request relation

Definition 11: *request relation, service role and request role.* Role r_i is called the request role of role r_j if role r_j provides the services requested by role r_i , i.e., $r_i.I\mathcal{M}_{out} \subseteq r_j.I\mathcal{M}_{in}$. A *request relation* denoted as Θ is a set of tuples $\langle r_i, r_j \rangle$, where r_i is a request role of r_j and r_j is a service role of r_i , i.e., $\langle r_i, r_j \rangle \in \Theta$ iff $\forall r_k, r_l \in \mathcal{R} (r_k.I\mathcal{M}_{out} \subseteq r_l.I\mathcal{M}_{in})$.

The request relation also holds the irreflexive property. This property guarantees a role does not issue a request itself in order to save the offset of message passing.

4.6 Derived relations

The following relations are derived from the above basic relations. They are not independent.

Definition 12: *competition relation and competitor role.* Role r_i is called a competitor role of role r_j if roles r_i and r_j have the same request role or the same upper role. A *competition relation* denoted as \mathcal{E} is a set of tuples of roles $\langle r_i, r_j \rangle$, where, r_i and r_j are competitor roles of each other. More exactly, $\langle r_i, r_j \rangle \in \mathcal{E}$ iff $\exists r_k \exists (\langle r_k, r_i \rangle \in \Theta \wedge \langle r_k, r_j \rangle \in \Theta) \vee (\langle r_i, r_k \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A})$.

For examples, roles *software consultant* and *software developers* are competitor roles because they provide same services, i.e., software development and role *client* may request the same service to them; in Figure 2, roles *researcher* and *assistant professor* are competitor roles but roles *programmer* and *researcher* are not competitor roles. The competition relations are used to establish regulations for assigning roles and dispatching messages.

Definition 13: *peer relation and peer role.* Role r_i is peer of role r_j if role r_i and r_j have the same supervisor role. A *peer relation* denoted as \diamond is a set of tuples of roles $\langle r_i, r_j \rangle$, where, r_i and r_j are peer roles of each other. More exactly, $\langle r_i, r_j \rangle \in \diamond$ iff $\exists r_k \exists (\langle r_i, r_k \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A})$.

4.7 Conflict relations

In role-based access control (RBAC), Nyanchama and Osborn point out that there are many conflicts when authorization of roles is concerned: user-user/group-group/user-group conflicts; role-role conflicts; privilege-privilege conflicts; user-role assignment conflicts; and role-privilege assignment conflicts. From the viewpoint of RBC, roles may be conflict when they are assigned to agents.

Definition 14: *conflict relation, conflict roles.* Roles r_i and r_j are conflict if one agent cannot play them together. r_i is called a conflict role of r_j and vice versa. A *conflict relation* denoted as Ξ is a set of tuples $\langle r_i, r_j \rangle$, where r_i, r_j are conflict roles of each other. More exactly, $\langle r_i, r_j \rangle \in \Xi \rightarrow \forall a \in \mathcal{A}, r_i, r_j \in \mathcal{R} (\neg (r_i \in a.\mathcal{R}_x \wedge r_j \in a.\mathcal{R}))$.

In a role-based system, the conflict relation is actually pre-defined in order to protect conflict roles from being assigned to the same agent.

4.8 Summary of role relations

As a summary, \mathcal{T} is used to express all the relations among roles in an RBC system. Suppose $r_i, r_j \in \mathcal{R}$, \mathcal{T} is a tuple with five binary relations and $\mathcal{T} ::= \langle \Omega, \mathcal{A}, \mathcal{A}, \Theta, \epsilon, \diamond, \Xi \rangle$, where,

- Ω : an inheritance relation. $\langle r_i, r_j \rangle \in \Omega$ if r_i inherits from r_j .
- \mathcal{A} : a promotion relation. $\langle r_i, r_j \rangle \in \mathcal{A}$ if r_i is a lower role of r_j and r_j is a upper role of r_i .
- \mathcal{A} : a report-to relations. $\langle r_i, r_j \rangle \in \mathcal{A}$ if r_i is a supervisee role of r_j and r_j is a supervisor role of r_i .
- Θ : a request relation. $\langle r_i, r_j \rangle \in \Theta$ if $r_i.I.\mathcal{M}_{out} \subseteq r_j.I.\mathcal{M}_{in}$.
- ϵ : a competition relation. $\langle r_i, r_j \rangle \in \epsilon$ if $\exists r_k \exists (\langle r_i, r_k \rangle \in \Theta \wedge \langle r_j, r_k \rangle \in \Theta) \vee (\langle r_i, r_k \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A})$.

- \diamond : a peer relation. $\langle r_i, r_j \rangle \in \diamond$ if $\exists r_k \exists (\langle r_i, r_k \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A})$.
- Ξ : a conflict relation. $\langle r_i, r_j \rangle \in \Xi$ if r_i and r_j are conflict.

Property 1: *irreflexive.*

- $\forall r_i \in \mathcal{R} (\langle r_i, r_i \rangle \notin \Omega)$;
- $\forall r_i \in \mathcal{R} (\langle r_i, r_i \rangle \notin \mathcal{A})$;
- $\forall r_i \in \mathcal{R} (\langle r_i, r_i \rangle \notin \mathcal{A})$; and
- $\forall r_i \in \mathcal{R} (\langle r_i, r_i \rangle \notin \Theta)$.

Property 2: *transitive.*

- $\forall r_i, r_j, r_k \in \mathcal{R} (\langle r_i, r_i \rangle \in \Omega \wedge \langle r_j, r_k \rangle \in \Omega \rightarrow \langle r_i, r_k \rangle \in \Omega)$;
- $\forall r_i, r_j, r_k \in \mathcal{R} (\langle r_i, r_i \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A} \rightarrow \langle r_i, r_k \rangle \in \mathcal{A})$; and
- $\forall r_i, r_j, r_k \in \mathcal{R} (\langle r_i, r_i \rangle \in \Delta \wedge \langle r_j, r_k \rangle \in \Delta \rightarrow \langle r_i, r_k \rangle \in \Delta)$;
- $\forall r_i, r_j, r_k \in \mathcal{R} (\langle r_i, r_i \rangle \in \epsilon \wedge \langle r_j, r_k \rangle \in \epsilon \rightarrow \langle r_i, r_k \rangle \in \epsilon)$; and
- $\forall r_i, r_j, r_k \in \mathcal{R} (\langle r_i, r_i \rangle \in \diamond \wedge \langle r_j, r_k \rangle \in \diamond \rightarrow \langle r_i, r_k \rangle \in \diamond)$.

Property 3: *symmetrical.*

- $\forall r_i, r_j \in \mathcal{R} (\langle r_i, r_j \rangle \in \epsilon \rightarrow \langle r_j, r_i \rangle \in \epsilon)$;
- $\forall r_i, r_j \in \mathcal{R} (\langle r_i, r_j \rangle \in \diamond \rightarrow \langle r_j, r_i \rangle \in \diamond)$; and
- $\forall r_i, r_j \in \mathcal{R} (\langle r_i, r_j \rangle \in \Xi \rightarrow \langle r_j, r_i \rangle \in \Xi)$.

Property 4: *noncircular.* There are not circles in inheritance, request, promotion, and report-to relations, i.e., $\neg \exists r_0, r_1, r_2, \dots, r_n \in \mathcal{R}, \langle r_i, r_{i+1} \rangle (r_i \neq r_{i+1}) \in \mathcal{X} (\mathcal{X} = \Omega, \mathcal{A}, \mathcal{A} \text{ or } \Theta) (i = 0, 1, \dots, n-1) \exists (r_n = r_0)$.

Definition 15: *role graph.* A role net is a directed graph [13] formed by all the relations of \mathcal{T} , denoted as $\mathcal{T.G}$, i.e., $\mathcal{T.G} ::= \langle \mathcal{R}, \mathcal{T} \rangle$, where, \mathcal{R} are the node set and \mathcal{T} are the edge set.

Definition 16: *interrelated roles.* Roles r_i and r_j are interrelated if $\langle r_i, r_j \rangle$ belongs to the role graph, i.e. $\langle r_i, r_j \rangle \in (\Omega \cup \mathcal{A} \cup \mathcal{A} \cup \Theta \cup \epsilon \cup \diamond \cup \Xi)$.

5. THE RELATIONS BETWEEN ROLES AND AGENTS

After roles and role relations have been set, a collaborative platform is built. Agents (actors) could collaborate (perform) on this platform (stage). Agents need to be assigned roles, behave on the platform, transfer roles and leave the platform. Therefore, the relations between agents and roles should be established. Some relations are interrelated and some are established by predicates.

Definition 17: *current role/agent.* Role r_i is the current role of agent a if a is currently playing r_i , i.e., $r_i = a.r_i$; at the same time, a is called as a current agent of role r_i , i.e., $a \in r_i.\mathcal{A}_c$.

Definition 18: potential role/agent. Role r_i is a potential role of agent a if a is qualified to play but not currently playing this role, i.e., $r_i \in a.R_p$; at the same time, a is called as a potential agent of r_i , i.e., $a \in r_i.A_p$.

From these definitions, it is clear to see that one agent can only play one current role but have many potential roles, and one role may have many current and potential agents.

Definition 19: past role/agent. Role r_i is a past role of agent a if a used to play r_i but r_i is neither a current nor a potential role, i.e., $r_i \in a.R_o$; at the same time, a is called as a past agent of r_i , i.e., $a \in r_i.A_o$.

Definition 20: apply-for (a, r_i). The predicate is to have agent a apply for role r_i .

After the predicate *apply-for* (a, r_i) is received, the system should check if a is qualified to play r_i based on the qualifications of it.

Definition 21: approve(a, r_i). It is a predicate that approves role r_i for agent a . This event occurs when *apply-for* (a, r_i) has been issued and the agent is evaluated to be qualified to play role r_i . After this predicated is executed, r_i is added to the potential agent set of agent a , i.e., $r_i \in a.R_p$.

Many agents may bid for one role. There must be a set of rules to select (approve) the best agent to play the role. It means that the approved agent should be the best one to accomplish the tasks specified by the role the highest efficiently and the least costly. Some system restrictions should be enforced when approving an agent: it can play a direct upper role of its repository roles, a leaf role, a lower role of its repository roles, a peer role and a past role but it cannot play a conflict role.

Definition 22: disapprove(a, r_i). It is a predicate that disapproves a role from an agent.

This predicates is executed in two situations: when an agent is promoted to play a upper role, this lower role is disapproved; when an agent is checked that its credits becomes below the required credit of the role, the agent is disapproved of this role. After it is executed, r_i is deleted from the repository set and put into the past role set of agent a , i.e., $r_i \notin a.R_x \wedge r_i \in R_o$.

Definition 23: transfer(a, r_i, r_j). It is a predicate that agent a transfers its current role from r_i to r_j . After it is executed, r_j is assigned to the current role of agent a and r_i is put back to the potential role set, i.e., $(a.r_c = r_j) \wedge (a.R_p = (a.R_p \cup \{r_i\}) - \{r_j\})$. It can be executed only when the group is still workable after the role is transferred.

Definition 24: dispatch(r_i, a, m). It is a predicate that role r_i dispatches message m to agent a . After it is executed, a does what the message asks for.

Definition 25: reply(a, r_i, p). It is a predicate that agent a replies an object p to role r_i . After it is executed, r_i

completes one request and responds to its request role with object p .

There are arguments that playing a role has two modes: parallel or sequential. Parallel playing means that an agent plays two or more roles at the same time and sequential playing means that an agent plays a role at a time. In RBC, an agent has only one current role and there are no parallel roles for one agent.

6. THE RELATIONS BETWEEN AGENTS

Many agents are working together in collaboration. By playing different roles, they build different workflows and accomplish different tasks, information transmissions, and productions.

In collaboration, competition is inevitable. Therefore, agents may collaborate or compete. Agents may be collaborators or competitors.

Definition 26: collaborator. Agent a and b are collaborators if their roles are interrelated, $\exists r_i, r_j \in a.R_d \cap b.R_x \ni \langle r_i, r_j \rangle \in (\Omega \cup \mathcal{A} \cup \mathcal{I} \cup \Theta \cup \mathcal{E} \cup \mathcal{D} \cup \Xi)$.

Definition 27: current collaborator. Agent a and agent b are current collaborators if they are currently playing interrelated roles or $a.r_c$ and $b.r_c$ are interrelated, i.e., $\langle a.r_c, b.r_c \rangle \in (\Omega \cup \mathcal{A} \cup \mathcal{I} \cup \Theta \cup \mathcal{E} \cup \mathcal{D} \cup \Xi)$.

Definition 28: potential collaborator. Agent a and b are potential collaborators if they are not current collaborators but there are two interrelated roles in the intersection of their repository role sets, i.e., $\exists r_i, r_j \in a.R_d \cap b.R_x \ni \langle r_i, r_j \rangle \in (\Omega \cup \mathcal{A} \cup \mathcal{I} \cup \Theta \cup \mathcal{E} \cup \mathcal{D} \cup \Xi) \wedge \langle a.r_c, b.r_c \rangle \in (\Omega \cup \mathcal{A} \cup \mathcal{I} \cup \Theta \cup \mathcal{E} \cup \mathcal{D} \cup \Xi)$.

Definition 29: peer. Agents a and b are peers if they play the same role, i.e., $a.R_c \cap b.R_c \neq \Phi$.

Definition 30: current peer. Agents a and b are current peers if they have the same current role, i.e., $a.r_c = b.r_c$.

Definition 31: potential peer. Agents a and b are potential peers if $a.R_d \cap b.R_x \neq \Phi \wedge a.r_c \neq b.r_c$.

Definition 32: competitors. Agents a and b are competitors if their repository role sets contain competition roles, i.e., $\exists r_i \in a.R_x, r_j \in b.R_x \ni \langle r_i, r_j \rangle \in \mathcal{E}$.

Definition 33: current competitor. Agents a and b are current competitors if their current roles are competitor roles, i.e., $\langle a.r_c, b.r_c \rangle \in \mathcal{E}$.

Definition 34: potential competitor. Agents a and b are potential competitors if $\exists r_i, r_j \in a.R_d \cap b.R_x \ni \langle r_i, r_j \rangle \in \mathcal{E} \wedge \langle a.r_c, b.r_c \rangle \notin \mathcal{E}$.

Definition 35: client and server. Agent a is called a client of agent b and b is called a server of a if $\exists r_i, r_j \in a.R_d \cap b.R_x \ni \langle r_i, r_j \rangle \in \Theta$.

Definition 36: current client and server. Agent a is called a current client of agent b and b is called a current

server of a if a is currently playing a request role of the role b is currently playing, i.e., $\langle a.r_o, b.r_c \rangle \in \Theta$.

Definition 37: *potential client and server.* Agent a is called a potential client of agent b and b is called a potential server of a if $\exists r_i, r_j \in a.R_d \cap b.R_r \exists \langle r_i, r_j \rangle \in \Theta \wedge \langle a.r_o, b.r_c \rangle \notin \Theta$.

7. PROPERTIES OF AN RBC SYSTEM AND THEIR APPLICATIONS

From the definitions 13, 26, 29 and 32, some interesting facts are presented: a) agents playing conflict roles are collaborators; b) peers are neither collaborators nor competitors; and c) peer role players are collaborators.

For a), it is true for sales context, roles *price setter* and *buyer* of a specific item are conflict but two persons playing these two roles are collaborators. In fact, two different agents are allowed and required sometimes to play different roles even though they are conflict.

For b), it is true in the context of peer reviewers in scientific research. Therefore, peers, collaborators and competitors are not interleaved as shown in Figure 4. There might be an argument that agents are both collaborators and competitors when they are playing the same role. However, from the above formal definitions, it is better to say that such agents are peers.

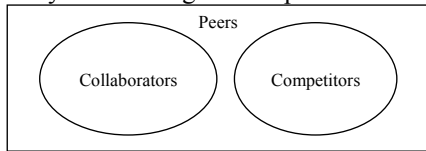


Figure 4. Collaborators, competitors, and peers

For c), it is true in the context of office work. The people in an office collaborate to complete a task ordered by their common supervisor.

Definition 38: *redundancy.* A system is redundant if some services are not requested, i.e., $\exists (r_j \in \mathcal{R})$

$$\exists (r_i.I.M_{in} \cup_{i=0}^{n-1} r_i.I.M_{out} (r_i \neq r_j)) \neq \Phi, \text{ where, } n = |\mathcal{R}|.$$

Definition 39: *insufficiency.* A system is insufficient if some requests are not served, i.e., $\exists (r_j \in \mathcal{R})$

$$\exists (r_i.I.M_{out} \cup_{i=0}^{n-1} r_i.I.M_{in} (r_i \neq r_j)) \neq \Phi, \text{ where, } n = |\mathcal{R}|.$$

Definition 40: *consistency.* A system is consistent if all the relations are consistent, i.e., they all keep the **Properties 1-4.**

One purpose of specifying role relations is to regulate message dispatches. The followings are some regulations:

- A message sent to a super role should be dispatched to its sub roles.
- A message accepted by an agent can be repackaged and forwarded to the supervisee roles of the role currently played by this agent.

- A message accepted by an agent can be repackaged and forwarded to the service roles of the role currently played by this agent.
- A message dispatched to an agent can be re-dispatched to the peers of this agent.

The relations and the properties specified in this paper have been applied in a prototype of role-based chatting system [15]. The success of the prototype verified the discussed relations.

8. RELATED RESEARCH

Although roles as concepts have been discussed for many years, the literature lacks systematic descriptions of the relations among roles and their players. Most of them did not present role relationship specification and classification. References discussed here mention role relationships and activate the author's work in this paper.

Marwell and Hage [6] present 100 role relationships in different societies including economic, political, health and welfare, science and education, family, religion, art and leisure sectors. Each role-relationship involves occupants, activities, locations and occurrences. All the relationships are classified in the senses of sociology, i.e., *gemeinshaft* and *gesellschaft*. Their analysis lacks abstraction in the sense of collaboration, management and systems and is not very useful in such fields.

In RBAC, Simon and Zurko [11] mention the role conflict relations when they discuss conflicts of interests relevant to complex tasks in a workflow management system. Conflicts common in business transactions to require two signatures before a check is issued, or some like measure to avoid fraud. Nyanchama and Osborn [7] discuss the role-role conflict relations by role graphs. A role-role conflict means that the two roles should never appear together. This implies they should never be assigned to a single user, which is checked on user-role assignment. Their discussion concentrates on the conflict of permission authorizations to access system resources.

Skarmas [12] proposes a role model to deal with the tree hierarchy of roles in organizations. Roles may contain other roles and/or elementary roles. In their proposed tree structures, elementary roles are leaves. In describing organizational relations, he introduces virtual roles to deal with the supervision relations among roles. Virtual roles are used to assign roles supervised by them to individuals. He mentions the role consistency problem mainly when role transfers occur. The problem in his model is that tasks of a role and roles themselves are not clearly separated.

Olarnsakul and Batanov [9] emphasize that roles have relationships (i.e., interactions and dependencies) with one another in order to fulfill assigned responsibilities. They present that roles and relationships are the building blocks of the organizational structure. In

their model, a role relationship is a set of protocols that represent collaboration tasks or business processes.

Odell et al. [8] describe a metamodel for agents, roles, and groups which are incorporated with Unified Modelling Language (UML) notations. They believe that roles provide the building blocks for agent social systems. Roles meet the requirements of describing interactions among agents. They use associations to describe the relationships among roles. Classifier classes are introduced to illustrate the relationships among agents and roles. Their metamodel is a good reference to analyze and design systems with roles.

Ren et al. [10] present a coordination model, i.e., the Actor, Role and Coordinator (ARC) model. Roles are taken as a key thrust in the ARC model. In their model, behaviors of actors, roles and coordinators are formally defined and applied into supporting the reconfigurability and fault localization for open distributed embedded software systems. The specified behaviors regulate some relations in role-based systems, such as, memberships and configurations.

9. CONCLUSIONS

A role engine is a platform to support RBC. The basic functions of a role engine are clarified and the fundamental relations in RBC are formally defined. These definitions can be used to find more properties and analyze the design and construction of an RBC system. An RBC system is very complicated. To check and keep it consistent is a hard work. The formal definitions of relations would help check its consistency.

To build a workable role engine, there are still many required researches and practices. To fully accept the RBC principles, one needs to accept that in a society it is the system structure form by its roles that controls the people's behavior. A role engine is just such a system to control agents' behavior. To make a role engine reality, it is needed to answer the following questions: Are the relations presented complete? Should the role engine or roles collaborate to manage the dispatches of messages? What is the number relation among roles and agents? Should time and space factors be introduced into roles?

ACKNOWLEDGMENTS

This research is in part supported by National Scientific and Engineering Research Council, Canada (NSERC: 262075-06) and IBM Eclipse Innovation Grant.

REFERENCES

- [1] Ashforth, B. E. *Role Transitions in Organizational Life: An Identity-based Perspective*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2001.
- [2] Black, J.S., "Work Role Transitions", *J. of International Business Studies*, vol. 19, no. 2, 1988, pp. 277-294.
- [3] Bostrom, R. P., "Role Conflict and Ambiguity: Critical Variables in the MIS User-Designer Relationship", *Proc. of the 17th Annual Computer Personnel Research Conference*, Miami, Florida, USA, 1980, pp. 88-115.
- [4] Colman, A. and Han, J., "Using role-based coordination to achieve software adaptability", *Science of Computer Programming*, vol. 64, 2007, pp. 223-245.
- [5] Kollingbaum, M., Norman, T., Mehandjiev, N., Brown, K., "Engineering Organization-Oriented Software", *WISER '06*, May 2006, Shanghai, China, pp. 23-28.
- [6] Marwell, G. and Hage, J., "The Organization of Role-Relationships: A Systematic Description", *American Sociological Review*, vol. 35, no. 5, 1970, pp. 884-900.
- [7] Nyanchama, M. and Osborn, S., "The Role Graph Model and Conflict of Interest", *ACM Trans. on Information and System Security*, vol. 2, no. 1, 1999, pp. 3-33.
- [8] Odell, J., Nodine, M. and Levy, R., "A Metamodel for Agents, Roles, and Groups", Odell, J., Giorgini, P., Müller, J. (Ed.), *Agent-Oriented Software Engineering (AOSE), Lecture Notes on Computer Science*, vol. 3382, Springer, Berlin, 2005, pp. 78-92.
- [9] Olarnsakul, M. and Batanov, D.N., "Customizing Component-Based Software Using Component Coordination Model", *Int'l J. of Software Engineering and Knowledge Engineering*, vol. 14, no. 2, 2004, pp. 103-140.
- [10] Ren, S., Yu, Y., Chen, N., Tsai, J. J.-P., and Kwiat, K., "The role of roles in supporting reconfigurability and fault localizations for open distributed and embedded systems", *ACM Trans. Autonomic & Adaptive Sys.*, vol. 2, no. 3, Sept. 2007, pp.10:1-27.
- [11] Simon, R. and Zurko, M. E., "Separation of duty in role based access control environments". *Proc. of the 10th IEEE Workshop on Computer Security Foundations*, Rockport, MA, June, 1997, Los Alamitos, CA, USA., pp. 183-194.
- [12] Skarmas, N., "Organizations through Roles and Agents", *Int'l Workshop on the Design of Cooperative Systems (Coop' 95)*, Antibes-Juan-Les-Pins, France, Jan. 1995, avail: <http://citeseer.ist.psu.edu/174774.html>.
- [13] Skiena, S., *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Reading, MA: Addison-Wesley, 1990.
- [14] Zhu, H., "Role as Dynamics of Agents in Multi-Agent Systems", *System and Informatics Science Notes*, vol. 1, no. 2, July 2007, pp. 165-171.
- [15] Zhu, H. and Alkins, R., "A Tool for Role-Based Chatting", *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, Montreal, Canada, Oct. 7-10, 2007, pp.3795-3800.
- [16] Zhu, H., Grenier, M., Alkins, R., Lacarte, J. and Hoskins, J., "A Visualized Tool for Role Transfer", submitted to *The 6th Int'l Conf. on Infor. Sys. for Crisis Response and Management*, Washington, D.C., USA, May 4-7, 2008.
- [17] Zhu, H. and Zhou, M.C., "Role-Based Collaboration and its Kernel Mechanisms", *IEEE Trans. on Systems, Man and Cybernetics, Part C*, vol. 36, no. 4, July 2006, pp. 578-589.
- [18] Zhu, H. and Zhou, M.C., "The Role Transferability in Emergency Management Systems", *Proc. of the 3rd Int'l Conf. on Info. Sys. for Crisis Response and Manag.(ISCRAM'06)*, Newark, USA, 2006, pp. 487-496.