East-West J. of Mathematics: Vol. 12, No 2 (2010) pp. 141-151

MODELING ROLES FOR ORGANIZATION IN OPEN DISTRIBUTED MULTIAGENT SYSTEMS

Dinh Que Tran[*] and **Manh Son Nguyen**[†]

[*]*Department of Information Technology*
Post and Telecommunication Institute of Technology,
Km10, Nguyen Trai, Ha Dong, Ha Noi - Vietnam
e-mail: tdque@ptithcm.edu.vn

[†]*Department of Information Technology*
Post and Telecommunication Institute of Technology,
Km10, Nguyen Trai, Ha Dong, Ha Noi - Vietnam
e-mail: manhsoncntt@yahoo.com

Abstract

Role has become an essential concept for designing interactions among agents in autonomous open distributed systems. While roles in closed distributed systems are completely defined during designing, roles in open multi-agent systems may be dynamically allocated during interaction between agents and depend on capabilities, actions, goals and types of organization that the agent is dispatched.

The purpose of this paper firstly extends the role approach with capabilities and actions and then develops a relational algebra among types of roles. An algorithm of allocating roles to agents and an application scenario corresponding to a social organization structure of a multi-auction commercial service system will be investigated.

1. Introduction

In the recent years, designing open multi agent systems based on the concept of role in social organization has attracted many researches ([1]-[18]). In contrast to closed multi-agent systems, whose roles are completely defined during

Key words: Multiagent system, modeling role, system design, social organization structure.
2000 AMS Mathematics Subject Classification: 30D30

designing and typically represented as a group of fixed tasks and responsibilities; in open multi-agent systems, roles may be dynamically allocated during interaction process between agents. It means that an agent which takes or does not take a role depending on its capabilities and goals the agent is dispatched. Various approaches to modeling roles, defining relationships among roles and assigning roles are generally based on some aspects such as capabilities, actions, goals, group and team and depended on social organization structure of systems. According to Deloach ([5], [6]), a role requires a set of capabilities and an agent undertaking the role needs to be assessed by degree function with value from 0 to 1. However, concrete properties of a role, behaviors assigning to the agent with the role as well as relation types are not described in this model. Odell et al. ([10], [11], [12]) attach role with the concept of group and then role is defined as a set of standardized responsibilities for behaviors in a defined group. Representing role and designing systems based on role in this model are expanded from UML and as a result, it becomes popular with community of object oriented development. However, like the model given by DeLoach, properties of role and behaviors agents need to perform when undertaking role are not defined. Xu and Zhang ([13], [14]) introduce a representation of role based on destinations, schedule, actions, a set of authorization, protocol and implemented state. Destination is attached to concrete roles and is carried out according to schedule set of that role. Relationship among roles is defined based on relations such as inheritance, incompatibility, composite etc. This model discriminates concepts of role and role instances to support role assignment in a system. According to Cabri et al ([1]- [4]), a role is represented as a set of capabilities and behaviors. Then these capabilities and behaviors will be dispatched to an agent undertaking the role. Interaction among agents with roles is described in BRAIN framework which in turn is based on actions and events; roles will be assigned during interaction process by RoleLoader. One of important features of this approach is that modeling role with starting requirements, capabiltites and behaviors are described in XML. This model, briefly B-role in this paper, is investigated and utilized in developing a relational algebra among roles and then building an algorithm of allocating roles to agents. The remainder of this paper is organized as follows. Section 2 considers the scenario of multi-auction in e-commerce which is used as illustration example and then presents formalization of roles and their relationships. Section 3 presents the algorithm of dynamically allocating roles to agents. Section 4 is a comparison with related works. Finally, Section 5 is conclusions and further work.

Table 1: Roles in multiauction commercial system

Manager	Manage the system
Notifier	Notify the results of auction section
Auctioneer	Seller with auction
Bidder	Buyer with auction
EnglishAuctioner	Seller with English auction
DutchAuctioner	Seller with Dutch auction
SealbidAuctioner	Seller with Sealbid auction
EnglishBidder	Buyer with English auction
DutchBidder	Buyer with Dutch auction
SealbidBidder	Buyer with Sealbid auction

2. Modeling Role and Role Relational Algebra

2.1. A Scenario: Multi-auction in E-Commerce

This section presents a scenario on multi-auction in commerce that is used as an illustration example for the rest of the paper. In such a context, buyer agents, on behalf of buyers, may select auction types such as English Auction, Dutch Auction, Sealed-bid Auction or Reserve Auction for purchasing. They can take more than one role related to various auction types. Seller agents, on behalf of sellers, may propose to sell items with different auction types. The system is managed by a Manager Agent whose responsibility is to control a role library and to assign roles to the corresponding agents. Roles in this system can be summarized as in Table 1.

2.2. Modeling Role

Definition 1. ([4]) *A role is defined as a tuple* $r =< Q, C, B >$*, in which* Q *is a set of initial requirements,* C *is a set of capabilities and* B *is a set of behaviors an agent undertakes when assigned to the role.*

In the multi-auction commercial system, the role English-Bidder is represented with $< Q, C, B >$ as in Table 2 and then in XML as in Table 3 in which three new XML tags are added: *Require:* a set of starting requirements for this role; *Capability:* a set of capabilities granted by the role agent; *Behavior:* a set of behavior for this role.

2.3. Relational algebra of Roles

Let $R = \{r_i\}$ be a set of roles, in which $r_i =< Q_i, C_i, B_i >$, i = 1,..,n are roles as in Definition 1.

Table 2: English bidder role

Role Name	**English Bidder**
Q	Own an account, know English auction rule
C	Join an English auction, re-bid
B	Bid, re-bid, payment

Definition 2. (Equality Relation) *Given two roles* $r_1 =< Q_1, C_1, B_1 >$ *and* $r_2 =< Q_2, C_2, B_2 >$. *Two roles are equal, denoted* $r_1 \equiv r_2$, *iff* $Q_1 = Q_2$, $C_1 = C_2$, $B_1 = B_2$.

Equality relation between two roles r_1 and r_2 introduces that when two agents fulfills all initial requirements, and has similar capabilities, they may own the same behaviors.

Proposition 1. *Equality relation* $\equiv$ *satisfies the following properties:*
(i) Reflexive: $\forall r \in R(r \equiv r)$
(ii) Symmetric: $\forall r_1, r_2 \in R(r_1 \equiv r_2 \Rightarrow r2 \equiv r1)$
(iii) Transitive: $\forall r_1, r_2, r_3 \in R((r_1 \equiv r_2) \wedge (r_2 \equiv r_3) \Rightarrow (r_1 \equiv r_3))$

Definition 3. (Inheritance Relation) *Given two roles* $r_1 = < Q_1, C_1, B_1 >$ *and* $r_2 = < Q_2, C_2, B_2 >$, *role* r_1 *inherits from role* r_2, *denoted* r_1 *extend* r_2, *iff* $Q_2 \subseteq Q1, C2 \subseteq C1, B2 \subseteq B1$.

Proposition 2. *Inheritance relation (extend) satisfies the following properties:*
(i) Reflexive: $\forall r \in R \Rightarrow (\ r$ *extend* $r)$
(ii) Transitive: $\forall r_1,\ r_2,\ r_3\ \in R((r_1$ *extend* $r_2) \wedge (r_2$ *extend* $r_3) \Rightarrow (r_1$ *extend* $r_3))$

In the multi-auction system, roles English-Bidder, Dutch-Bidder inherit from Bidder; roles English-Auctioner, Dutch-Auctioner inherit from Auctioner. And then, we can consider Bidder and Auctioner as abstract roles and inheritance relation can be represented as in Table 4.

Definition 4. (Incompactibility Relation): *Given two role* $r_1 =< Q_1, C_1, B_1 >$ *and* $r_2 =< Q_2, C_2, B_2 >$, r_1 *is incompactible w.r.t* r_2, *denoted* r_1 *not* r_2, *iff* $Q_1 \cap Q_2 = \emptyset$

Incompactibility relation between two role r_1 and r_2 shows that they cannot be taken by an agent at the same time. Let Incomp(r_1) to be a set of incompactible roles w.r.t r_1. It is easy to prove the following propostition:

Proposition 3. *Incompacitability relation satisfies the following properties:*
(i) Non-reflexive: $\forall r \in R(r$ *not* $r)$

Table 3: English bidder role in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<role xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                              xsi:noNamespaceSchemaLocation="XRole.xsd">
        <name>bidder</name>
        <context>auction</context>
        <description>
                This-role-is-the-bidder-of-an-auction.
        </description>
                <keyword>auction</keyword>
                <keyword>bidder</keyword>
                        ...
        <require>
                <name>account</name>
                <description>has-a-valid-account-to-bid</description>
        </require>

        <capability>
                <name>join</name>
                <description>join-an-auction</description>
        </capability>
        <capability>
                <name>re-bid</name>
                <description>make-more-than-one-bid</description>
        </capability>
                        .
        <behavior>
                <name>bid</name>
                <description>Makes-a-bid.</description>
                <content>
                        <description>Bid.</description>
                        <type>Price</type>
                </content>
        </behavior>
                ...
</role>
```

Table 4: Inheritance Relation between English-Bidder and Bidder

Role name	**Bidder**	**English Bidder**
Q	Own an account	own an account, know English auction rule
C	re-bid	re-bid, join an English auction
B	bid, rebid, payment	bid, re-bid, payment

Table 5: Including relation between Manager and Notifier

Role name	**Manager**	**Notifier**
Q	own an admin account	own an admin account
C	creat an auction, assign role	Notify
B	create, assign role	Notify

(ii) Symmetric: $\forall r_1,\ r_2 \in\ R\ (r_1\ not\ r_2 \Rightarrow r_2\ not\ r_1)$

Definition 5. (Including relation) *Given two roles* $r_1 =< Q_1, C_1, B_1 >$ *and* $r_2 =< Q_2, C_2, B_2 >$, r_1 *is including* r_2, *denoted* r_1 *includes* r_2, *iff* $Q_2 \subseteq Q_1$.

Definition 6. (Incompactibility Relation): *Given two role* $r_1 =< Q_1, C_1, B_1 >$ *and* $r_2 =< Q_2, C_2, B_2 >$, r_1 *is incompactible w.r.t* r_2, *denoted* r_1 *not* r_2, *iff* $Q_1 \cap Q_2 = \emptyset$

Including relation between two roles r_1 and role r_2 shows that when an agent takes the role r_1, it must also be assigned r_2.

Proposition 4. *Including relation satisfies the following properties:*
(i) Reflexive: $\forall r \in R(r\ and\ r)$
(ii) Transitive: $\forall(r_1,\ r_2,\ r_3) \in R((r_1\ and\ r_2) \wedge (r_2\ and\ r_3) \Rightarrow (r_1\ and\ r_3))$

In multi-auction commercial system, roles Manager and Notifier have including relation (agent which takes Manager is also assigned role Notifier). Table 5 represents Manager and Notifier with $< Q, C, B >$.

Definition 7. *(Add-in relation) Given two roles* $r_1 =< Q_1, C_1, B_1 >$ *and* $r_2 =< Q_2, C_2, B_2 >$, r_1 *is add-in w.r.t.* $r2$, *denoted* r_1 *add* r_2, *iff* r_1 *Incomp(*r_2*) and* $B_1 \cap B_2 = \emptyset$.

Add-in relation between two roles shows that both of them can be assigned to an agent at the same time and then sets of behaviors corresponding to two

roles are separate.

Proposition 5. *Add-in relation satisfies two properties:*
(i) Reflexive: $\forall r \in R$ *(r add r)*
(ii) Symmetric: $\forall r_1, r_2 \in R$, *(*r_1 *add* $r_2 \Rightarrow r_2$ *add* r_1*)*

In multi-auction commercial system, roles Bidder and Auctioner has add-in relation. For example, a buyer agent may join more than one auction-section, and a seller may take part in a different section with different items.

3. Allocating Roles to Agents (ARA) - An Algorithm

This section introduces the algorithm ARA (ARA: Allocating Roles to Agents) of allocating role. Let R be a set of roles, each of which is represented in the form $< Q, C, B >$ and relations on R are defined as in Section 2. Consider the system with the following social organizational characteristics: Leading role is a special role that is responsible for managing the role library and assigning roles to agents. Manager agent will undertake this role and then, all ordinary agents must register to the agent before taking roles. Being accepted, ordinary agents may send a request to Manager agent to take or release roles. There are three types of requests: *Query*: checking the availability of a proposed role; *TakeRole*: taking a role;*Release*: releasing a role.

Let R be a role class library represented in XML, $a[]$ a set of ordinary agents and $r*$ be a leading role.

The following operators are used in the role assigning algorithm.

CreatManagerAgent(agent): Creating a manager agent, assigning Manager role to the agent;

RegisterOrdinaryAgent(agent): Registering ordinary agents requiring multi-auction services;

TakeLeadingRole(am, r)*: Assigning Leader Role to Manager Agent and in turn, the agent is responsible to assign roles to ordinary agents;

CreatRoleInstances(am, R): Manager agent takes responsibilities to manage role library and role instances.

ReceiveRequest(a[i]): Receiving requests from ordinary agent a[i] and then returning a Request Type.

Reserve(r): Reserving the role r for the agent sending the Query Request

Notify(a[i], r, message): Notifying the agent a[i] about status Found and NotFound of role r.

Assign(a[i], r): Assigning role r to agent a[i], including a set of capabilities C and a set of behaviors B.

Table 6: ARA Algorithm

```
ARA-algorithm()
    1. Inititialize
        CreatManagerAgent(a_m);
        RegisterOrdinaryAgent(a[]);
    2. Create role instances
        TakeLeadingRole(a_m, r*);
        CreatRoleInstances(a_m, R);
    3. Assigning role
        for each agent a[i] in a[]
            Q = ReceiveRequest(a[i]);
            case Q is Query(r)
                if Search(r, R) then
                    Reserve(r);
                    Notify(a[i], r, Found);
                else
                    Notify(a[i], r, NotFound);
            case Q is TakeRole(r)
                if Check-Start-Requirement(a[i], r) then
                    Assign(a[i], r)
                if Check-Compatibility(a[i].role, r) then
                    Suspend(r.Behavior);
            case Q is Realease(r)
                Deact(a[i], r);
        endfor
        UpdateRelationship(a[]);
End ARA-algorithm
```

Check-Start-Requirement(a[i], r): Checking starting requirements against agent a[i] with role r. This function returns a Boolean value.

Check-Compactibility(a[i].role, r): Checking the compactability of the role r with other roles assigned to a[i]. Returning is a Boolean value.

Suspend(r.Behavior): Stoping all behaviors of agents whose roles are incompactable with role r.

Deact(a[i], r): Suspending execution of actions of role r by agent a[i].

4. Related Works

Modeling a role with a 3-tuple $< Q, C, B >$ in XML was introduced by Cabri et. al ([2], [3], [4]), but in their model, role relationship algebra is not fully considered. Xu and Zhang ([13], [14]) also considered some role relationships and proposed an role assignment algorithm AR-mapping. However, their relationships and algorithm were only suitable to their role model in Z specification and moreover components of role are not described clearly in relationships. Relations of role algebra, such as equal, merge, incompactibility etc., were considered by Karageogos et al. [9], but their role definition is based on responsibilities and the linking between components w.r.t. each relation is not defined. In this paper, we have used the role model with 3-tupe $< Q, C, B >$ to develop a role relational algebra with these components. Furthermore, based on these relations, we have constructed the ARA-algorithm for allocating roles to agents.

5. Conclusion

This paper presents the algebra of roles based on the role model with 3-tupe $< Q, C, B >$ proposed by Cabri et al. ([1]- [4]). In this relational structure, some relations such as equality, inheritance, incompactability, including, and add-in are considered as criteria for designing the algorithm of allocating roles (ARA) in a multiagent system. Manager Agent owns the "super-role" Leader that is responsible for managing and assigning roles to agents. Such an architecture introduces the characteristics of a social organizational structure for the role allocation algorithm. A scenario of application of these relations has been proposed for the multi-auction commercial system. We are currently investigating additional relations of roles as well as the process of allocation of roles within different social organization structures. We are also working on designing and implementing the architecture with roles and the application scenario of multi-auction in JADE Framework ([18]). The research results will be presented in our further work.

References

[1] G.Cabri, L. Ferrari and L. Leonardi, *Agent Role-based Collaboration and Coordination: A Survey about existing Approaches*, SMC (6), pp. 5473–5478(2004)

[2] G.Cabri, L. Leonardi, F. Zambonelli, *Role-based Interaction Infrastructures for Internet Agents*, IEEE TRANSACTIONS on Information and Systems, Vol. E86-D, No.11, pp. 2262–2270 (2003)

[3] G.Cabri, L. Leonardi, F. Zambonelli, *XRole: XML Roles for agent Interaction*, Proceedings of the second international joint conference on Autonomous agents and multiagent system, pp. 105–112 (2003)

[4] G.Cabri, L. Ferrari and L. Leonardi, *Rethinking agent roles: Extending the role definition in the BRAIN Framework*, International Conference on Systems, Man and Cybernetics, ISBN 0-7803-8566-7 (2004)

[5] S.A. DeLoach, *Engineering Organization-Based Multiagent System*, SELMAS 2005, pp.109-125 (2006)

[6] S.A. DeLoach, *Developing a Multiagent Conference Management System Using the O-MaSE Process Framework*, Proceedings of the 8th International Workshop on Agent Oriented Software Engineering, Hawaii (2007)

[7] M. Dastani, M.B.Riemsdijk, J.Hulstijin, J.Dignum, J.Meyer, *Enacting and Deacting Roles in Agent Programming*, In LNAI 3382, Proceedings of the Agent Oriented Software Engineering Workshop (AOSE'04), New York, pp.189-204 (2004)

[8] M.Dastani, V.Dignum, F.Dignum, *Role-Assignment in Open Agent Societies*, In Proceedings of AAMAS03, Second International Joint Conference on Autonomous Agents and Multi-agent Systems, Melbourne, Australia (2003)

[9] A.Karageogos, S.Thompson, N.Mehandjiev, *Specifying reuse concerns in agent system design using a role algebra*, In: Agent Technologies, Infrastructures, Tools, and Applications for e-Services, Lecture Notes in Artificial Intelligence LNAI, 2592. Springer-Verlag, ISBN 3-540-00742-3 (2003)

[10] J.Odell, H.Parunak, S.Brueckner, J.Sauter, *Changing roles: Dynamic role assignment*, Journal of Object Technology, ETH Zurich. V2, No. 5, pp. 77-86 (2003)

[11] J.Odell, M.Nodine M, R.Levy, *A Metamodel for Agents, Roles and Groups, in: Agent-Oriented Software Engineering*, AOSE (2004).

[12] J.Odell, H.Parunak, M.Fleischer, *The role of roles in designing effective agent organizations*, In Alessandro Garcia, C. Lucena, F. Zambonelli, A. Omicini, and J. Castro, editors, Software engineering for large-scale multiagent systems, Vol.2603 of Lecture Notes on Computer Science, pp 22-28, Springer, Berlin (2003)

[13] H.Xu, X.Zhang, *A Methodology for role-based modeling of open multi-agent software systems*, In Proceedings of the 7th International Conference on Enterprise Information Systems (ICEIS 2005), May 24-28, 2005, Miami, Florida, USA, pp. 246-253 (2005)

[14] H.Xu, X.Zhang, R. J. Patel, *Developing role-based open multi-agent software systems*, International Journal of Computational Intelligence Theory and Practice (IJCITP), June 2007, Vol. 2, No. 1, pp. 39-56 (2007)

[15] E.A. Kendall, *Agent roles and role models: New abstractions for intelligent agent systems analysis and design*, ECOOP'99, AOP Workshop, Lisbon, (1999)

[16] V.D.Dang, N.Jennings, *Optimal clearing algorithms for multi-unit single-item and multi-unit combinatorial auctions with demand/supply function bidding*, Proc. 5th Int. Conf. on Electronic Commerce, Pittsburgh, USA, 25-30 (2003)

[17] H.Ben-Ameur, B.Chaib-draa, P.Kropf, *Multi-item Auctions for Automatic Negotiation*, Journal of Information and Software Technology, 44:291–301 (2002)

[18] JADE http://jade.tilab.com/