Access Control for Cooperation Systems Based on Group Situation

Minsoo Kim[1], James B.D. Joshi[2], and Minkoo Kim[1]

[1] Graduate School of Information and Communication, Ajou University
Suwon, Korea (South)
{visual,minkoo}@ajou.ac.kr
[2] School of Inforamtion Science, University of Pittsburgh, PA, USA
jjoshi@mail.sis.pitt.edu

Abstract. Cooperation systems characterize many emerging environments such as ubiquitous and pervasive systems. Agent based cooperation systems have been proposed in the literature to address challenges of such emerging application environments. A key aspect of such agent based cooperation system is the group situation that changes dynamically and governs the requirements of the cooperation. While individual agent context is important, the overall cooperation behavior is more driven by the group context because of relationships and interactions between agents. Dynamic access control based on group situation is a crucial challenge in such cooperation systems. In this paper we propose a dynamic role based access control model for cooperation systems based on group situation. The model emphasizes capability based agent to role mapping and group situation based permission assignment to allow capturing dynamic access policies that evolve continuously.

Keywords: Group Situation, Access Control, Cooperation System.

1 Introduction

A cooperation system involves a group of diverse entities interacting with each other for achieving common goals. Cooperation based approaches have been recently being pursued as a promising way to build large and complex ubiquitous or pervasive application environments. Several cooperation based approaches such as Gaia [1] and community computing [2] have been proposed in the literature. These approaches focus on the design and implementation of cooperation systems in highly dynamic environments. In these approaches, the concept of role is used to define interactions among agents. This idea gives well-recognized advantages to system developers with regards to separation of concerns between algorithmic issues and the interaction issues, reuse of solution and experience, and dynamic system management of operation [3, 4].

Group situation based cooperation model was proposed as a model for designing cooperation using roles similar to approaches mentioned above by Kim et al. [5]. The main contribution of this model is to design interactions among roles by defining group situations. All agents perform their own action(s) and interaction(s) with others

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 11–23, 2009.

based on the current group situation and, eventually, the goal of the group is achieved by such a situation flow [5]. The group situation is expressed by aggregation of context of agents within the group.

Situation-awareness in systems is an emerging technology that goes beyond context-awareness that makes pervasive systems more dynamic [5, 19]. Context, which refers to any useful information for the operation of an application, has become one of the key requirements of current and emerging systems, such as web services, ubiquitous systems, and business enterprises [9]. Context-awareness in such application provides capability for fulfilling user and environmental requirements. Moreover, in systems composed of multiple entities, for example multi-agent systems, awareness of group situation as well as individual situation is very crucial. For instance, one agent can decide to perform a certain authorized action based on its context, but the overall group context may require the interruption of that action. In summary, the group situation based cooperation model emphasizes appropriate cooperation among agents to ensure that the cooperation goal is achieved.

Existing models do not address security issues and in particular access control requirements within such cooperation systems. Access control refers to the process of limiting access to the resources of a system only to authorized programs, processes, or other systems so as to ensure that confidentiality, integrity and availability requirements are met [10]. Within a cooperation system there can be two types of access control requirements for cooperation systems. One is an access control to ensure authorized access to resources and information shared by agents, the other is access control related to the authorized interactions among agents.

Role-Based Access Control (RBAC) [10] and its variations have been found to be the most promising approaches for addressing access control issues related to complex and emerging systems. RBAC approach is very desirable in terms of powerful functions and flexibility for addressing the security needs of systems and applications. Moreover, it provides a mechanism for reducing the complexity, time, and potential security faults within the organization. To avail of these advantages of RBAC, various systems such as large enterprises and web services are adopting RBAC. Recently, Attribute-Based Access Control (ABAC) [11][12], Rule-Based RBAC (RB-RBAC) [13] and Context-RBAC [14][15] models that support dynamic user-role and role-permission assignments using knowledge about users and systems have been proposed.

Access control in cooperation system should consider the group situation in addition to individual context. For example, one agent has permission to an object, but sometimes the group might limit the permission depending on group situation. The main contribution of this paper is towards addressing this idea; i.e. *establishing access control mechanism based on group situation in cooperation system*. In this paper, we propose a group situation driven RBAC (GS-RBAC) model, which has the following key distinguishing characteristics.

- *Automatic user-role mapping*. In the cooperation systems, roles are subjects that perform their actions and interact with others, so user-role mapping is a process which finds the best user who can play the role. Therefore, capability of user is an important property in the mapping process, and capability matching methods such as agent matchmaking can be used for mapping users to roles dynamically and automatically.

- *Dynamic role-permission assignment based on group situation*. On a given group situation, each agent that is mapped to role performs its own actions and/or and engages in cooperative interactions for achieving common goals. To ensure success of these actions and interactions, proper permissions are given to proper roles depending on group situation.
- *Constraints for user-role mapping are different from those for role-permission assignment*. The former is context-based constraints to find a best user who can play a role, while the latter is group situation based constraints to assign permissions to the role. While the mapped roles to users do not change during cooperation, permissions are dynamically changed depending on changes of group situation.

The remainders of the paper are structured as follows. In section 2 we describe motivating example, and in section 3 we introduce situation-related researches and the group situation based cooperation model. In section 4, we present our proposed GS-RBAC model. Illustrative examples are described in section 5, and finally the conclusions and future work are discussed in section 6.

2 A Motivating Example – *Preparing Presentation*

Scene 1. John, head of personnel section, receives directions from his boss. He has to prepare a presentation within next 8 hours about a performance rating of all the employees in the company. This will involve the following tasks - analyzing employee data, researching sections, creating statistics, and preparing a presentation document.

Scene 2. John selects members to form a cooperation group to prepare the presentation - two for analyzing employee data, one for researching sections, one for statistics for the ranking, and one member for preparing a presentation document. They cooperate with each others, works individually. John has to make a schedule - when members need to cooperate and when they work alone. Each member has daily regular work that he/she has to do.

Scene 3. Member m_1 is analyzing employee data by using an employee database. He interacts with member m_2 who is researching sections. Member m_3 needs to work on the statistics and to interact with m_1 and m_2, and can access the employee database after m_1 finishes using.

Cooperation is frequently required for solving complex problem in computer systems as well as in human society. As shown in above examples, there are several issues in cooperation. First, which tasks are needed for cooperation should be analyzed - *cooperation process* (scene 1) should be identified. This process indicates which agents can engage in what type of interaction with which other agents and when. Second, it needs to find agents that are engaged in the defined cooperation - which can be referred to as the *organizing process* (scene 2). Note that cooperation dynamically assigned tasks of agents. Therefore, it is important to find agents that can perform cooperative takes defined in a cooperation process. Finally, access control needs to be defined to ensure that agents engage in authorized interactions and make

only authorized access to resources shared among agents - which is basically the *access control process* (scene 3). Even though all the members in a group have permissions to access a resource, sometimes it needs to limit use of those permissions in order to ensure the success of cooperation.

3 Context, Situation, and Group Situation

3.1 Context and Situation

Attribute, context and situation are concepts representing knowledge about objects or systems and hence, at times it is difficult to clearly distinguish among them. Generally, an attribute is a specification that defines a property of an object, computing elements or system - e.g. size of file and age of human. Context is referred to as 'any information that can be used to characterize the situation of an entity (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and applications themselves'. [9]. Therefore, any attribute of a system can be a context and can be used for *manipulating* context. In other words, context is more conceptualized as a set of attributes. For example, aggregation of attributes *temperature* and *humidity* can manipulate context *weather*. We can say 'it is so hot' when temperature is 95F and humidity is 80%. Using context, the status of system can be represented in a more comprehensive form and made more human understandable.

The situation research has long history and now it has been re-focused to represent systems more comprehensively. Situation has several different definitions. First, McCarthy, the originator of the situation calculus, defines situation *'is the complete state of the universe at an instant of time'* [6]. Reiter formalized the situation calculus and argues situation '*is a finite sequence of actions. It's not a state, it's not a snapshot, it's a history'* [7]. In researches and systems which adopted McCarthy' definition, a situation is described by using properties of systems such as variables, attributes and context. In contrary, if a situation is a history, for example in GOLOG which is a programming language for describing situation calculus, the state of the world is described by functions and relations (fluent) relativized to a situation.

Recently, Stephan Yau et al. defines situation in the context of pervasive environment as follows [8]:

The situation of an application software system is an expression on previous device-action record over a period of time and/or the variation of a set of contexts relevant to the application software on the device over a period of time. Situation is used to trigger further device actions.

This definition seems to combine McCarthy and Reiter' definitions and provide a more comprehensive understanding. First, the concept of context which is one of major requirements of pervasive computing is inserted into situation. Second, they refer the purpose of situation which means to trigger the further actions of systems. Of course this is not a new, but it provides comprehensive way to researches for developing systems having ability of situation-awareness. However, this definition does not show what differences between context and situation are and what meaning (semantic) of situation is (situation is not just expression).

We define situation as follows:

Situation is a problematic state of a computational element. The problem is recognized by being aware context information, and it needs to plan (a sequence of actions) for solving the problem on the situation. Context information includes changes of context, relations between context as well as individual context.

Situation is a concept which includes planning as well as context awareness. Context information characterizes problem which has to be solved on a given situation, and a sequence of further actions are specified to solve the problem on the situation. Situation does not use to trigger actions, but the actions are triggered by recognizing a set of context information (context-awareness) and history of actions (planning). Therefore, situation-awareness means to recognize problem by context-awareness and to discover (plan) a sequence of actions for solving problem. One important difference between context and situation is that the former is domain dependent and the latter is system-dependent. In other words, situation is specified by each system while context may be shared by applications on same domain (sometimes on most systems). For example, 'hot' or 'temperature is 95F' may be understood as same meaning on most systems, but situation may be specified as different ways (different problems, different plans).

3.2 Group Situation Based Cooperation

To explain a group situation, we first define a cooperative group [5]. A group situation includes aggregation of individual context, relationship among those context, and relationship among individuals (agents). A group is a metaphor of cooperation system which is composed of multiple entities (agents), common goals, and a cooperation process. Moreover, when considering the dynamic environment such as a ubiquitous environment, dynamic group organization is required. Therefore we define the cooperative group including organization process in [5].

Definition of *group situation* is composed of two parts; context information and a sequence of cooperative actions. Context information means a status of agents, relation between agents, and changes of context within a group. Moreover it works as pre-conditions for cooperative actions with history of cooperation.

On a given situation s_i, each agent within a group performs actions listed in f_i. The result of agents' actions makes changes of group situation to s_j, and then agents perform actions in f_j. After all, a goal can be achieved by following group situations from s_0 to s_g (s_0 is a start situation when group organizes and s_g is a goal situation when cooperation ends). More detail description about group situation based cooperation model is described in [5]. Applications and examples using group situation based cooperation is shown in [2] with community computing paradigm which was proposed for building dynamic cooperation systems in ubiquitous environment.

4 Group Situation Based Access Control

The key idea behind group situation based cooperation model is that a group situation can be defined by aggregating agents' context and a cooperation process can be

defined for each group situation. However, this model does not consider the access control issues for ensuring security for shared resources among agents. We propose an access control method that controls accesses to resources and agents based on group situation so that the agents and resources involved in the cooperation are protected while the cooperation goals are achieved. Existing RBAC variations do not support group context and fine-grained situation awareness, especially for agent-based cooperation environments. In the proposed secure cooperation model, actions of agent required for cooperation is driven by group situation. That is, the permissions required for performing agents' actions and interactions (i.e. cooperation) are driven by group situation.

For the proposed GS-RBAC model, we define following sets by extending the standard RBAC model [10].

- USERS: a set of users or autonomous software/hardware agents.
- ROLES: a set of roles defined within a cooperative group.
- OBJS: a set of objects within a cooperative group
- OPS: a set of applicable operations on OBJS
- PERS: a set of permissions, PERS $\subseteq$ OBJS $\times$ OPS
- UM: a many-to-many user-to-role mapping, UM $\subseteq$ USERS $\times$ ROLES
- PA: a many-to-many permission-to-role assignment, PA $\subseteq$ PERS $\times$ OLES

Additionally, for user-role mapping there are three constraint sets.

- CAPS: a set of capability of roles and users
- CONTEXTS: a set of context of role and users
- QOSS: a Set of QoS parameters of capabilities

4.1 User-Role Mapping

In the group situation based cooperation model [5], the group organizing process can be considered as user-role assignment process in traditional RBAC. When organizing a group, however, all agents are not assigned to roles, but only best agents are mapped to roles. In other words, the agent-role mapping process is to find the best agents that can play a given role. This is similar to the agent match-making and service discovery processes. Three constraints are used for user-role mapping in our model.

Capability Matching. To map a user to a role, a user should have capabilities that are depicted in the role descriptions. Role description includes role's capabilities for cooperating with others, and the cooperation process. Therefore, an agent should be able to act or interact with others according to the cooperation process which is required by the cooperative group.

Context-based constraints. Only users that satisfy context-based constraints can be candidates for final mapping to a role. These constraints come into effect only when a group is organized. Note that role-mapping in the cooperation model is a group organizing process. Therefore, if a user is mapped to a role, then this relation is not changed until the cooperative group is dissolved.

QoS filtering. There may be many users having same capability in large scale environments, so several users can be candidates for assuming a role. Therefore, we need to select proper agents among them. QoS (Quality of Service) is a popular way to filter or prune bad ones during agent matchmaking and web service discovery. Hence, for role-mapping we adopt this method and use the following QoS parameters introduced in [17]: *agent trustworthiness*, *reliability*, *availability*, *robustness* of capabilities, and *response time*.

4.2 Role-Permission Assignment

Standard RBAC supports only static role-permission assignment, and hence is not adequate for the dynamic computing environment where permissions need to be assigned dynamically. More recent extended RBAC models have focused on capturing context information and hence support dynamic assignments.

A key aspect of the proposed model is that the permission assignment is dynamically driven by changes of group situations. Since objects within a group are shared among agents, conflicts and security problems can result during cooperation, if only static permission assignment is considered and proposed situation information is not incorporated in the access control policy. Even though several RBAC variations support dynamic role-permission assignment, they are not applicable for cooperation systems because only context of individual agent is considered for the assignment. Each agent works for cooperation by grasping status of others and group. Therefore role-permission assignment should support cooperation among agents. In order to do this, we propose role-permission assignment based on group situation. This means that the role-permission assignment should support all agents' actions on a given situation, because each agent takes actions as cooperation in group situation based cooperation model. All permissions that are required for performing cooperation are given to agents within the group in a given situation. Given the start situation s_0 and the goal situation s_g, while user-role mapping is done once in s_0, role-permission assignment is done in every intermediate group situation.

Table 1 lists status predicates of the group situation based access control model. First column describes predicates, second column specifies evaluation domain for predicates, and third column describes semantics of the predicates.

Predicate *u_mapped (u, r, c, s)* refers to user-role mapping in situation *s*, and *u_mapped (u,r,c,s)*, *u_mapped (u, r, c, x, s)*, and *u_mapped (u, r, c, x, q, s)* refer to user-role mapping with matching capability *c*, satisfying context-based constraints *x*, and filtering QoS parameter *q*. Each constraint can be a set, for example $u\_mapped\ (u_1, r_1, \{c_1, c_2,\}, \{x_1, x_2, x_3\}, q_1)$ indicates "user u_1 is mapped to r_1 with capabilities $\{c_1, c_2 \mid move(), compute()\}$, context-based constraints $\{x_1, x_2, x_3 \mid time=3p.m., age>30, location=inBuilding\}$, and QoS parameter $\{q_1, reliability>80\}$".

Now we introduce the following axioms, based on those earlier proposed by Joshi et al. in GTRBAC [20], to capture the key relationships among status predicates in Table 1. In our cooperation model, especially, user-role mapping is only accomplished in the starting situation s_0, and de-mapped in the goal situation s_g. This means that users mapped to roles can activate the roles in all situations (from s_0 to s_g).

Table 1. Status predicates of group situation based RBAC

Events	Predicates	Evaluation Domains	Semantics
P: set of permissions, R: set of role, U: set of users, C: set of capabilities, Q: set of QoS parameters, X: set of user context, S: set of group situation, $r \in R, p \in P, u \in U, c \subset C, q \subset Q, x \subset X, s \in S$			
Role Enabling	enabled (*r, s*)	R×S	*r* is enabled in *s*
User-Role Mapping	u_mapped (*u, r, s*)	U×R×S	*u* is mapped to *r* in *s*
	u_mapped (*u, r, c, s*)	U×R×C×S	*u* is mapped to *r* with *c* in *s* (*c* is a sub-set of *C*)
	u_ mapped (*u,r,c,x,s*)	U×R×C×X×S	*u* is mapped to *r* with *c* and *x* in *s* (*x* is a sub-set of *X*)
	u_ mapped (*u,r,c,x,q,s*)	U×R×C×X×Q×S	*u* is mapped to *r* with *c*, *x*, and *q* in *s* (*q* is a sub-set of *Q*)
Role-Permission Assignment	p_assigned (*p, r, s*)	P×R×S	*p* is assigned to *r* in *s*
Run-Time Events	can_activate (*u,r,s*)	U×R×S	*u* can activate *r* in *s*
	can_acquire (*u, p, s*)	U×P×S	*u* can acquire *p* in *s*
	r_can_acquire (*u,p,r,s*)	U×P×R×S	*u* can acquire *p* through *r* in *s*
	can_be_acquired (*p,r,s*)	P×R×S	*p* can be acquired through *r* in *s*
	acquires (*u,p,t*)	U×P×T	*u* acquires *p* in *s*
	r_acquires (*u,p,r,s*)	U×P×R×S	*u* acquires *p* through *r* in *s*

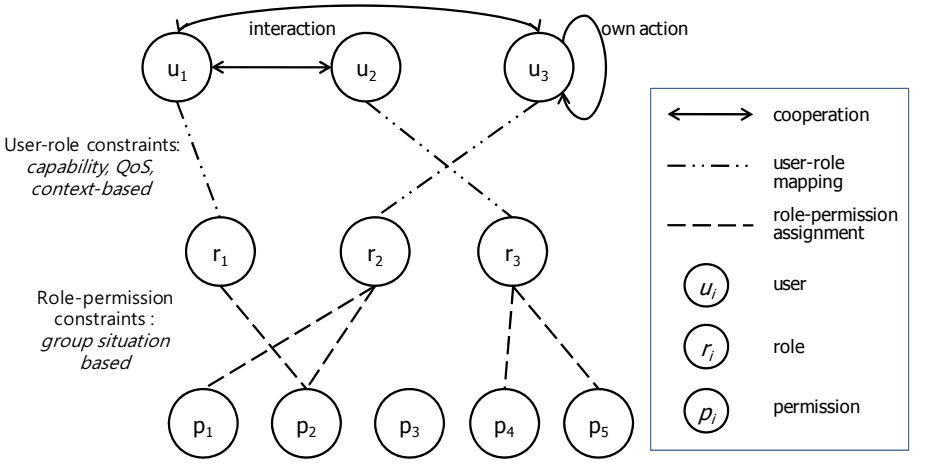

Fig. 1. Group situation based access control in cooperation system. In run time, users mapped to roles cooperate with each other according to the cooperation process described for roles. Capability matching, context-based constraints, QoS parameters are used for user-role mapping, and role-permission assignment is accomplished depending on changes of group situation.

Next axioms show semantic relations between status predicates.

Axioms. For all r $\in$ ROLES, $p \in$ PERS, $u \in$ USERS, $s \in$ SITUATIONS, the following implications hold:

1. u_mapped $(u,r,s) \rightarrow$ can_activate(u,r,s)
2. u_mapped $(u,r,s_0) \rightarrow$ can_activate (u,r,s_i), $i \in \{0, \ldots, g\}$, s_0 = starting situation, s_g=goal situation
3. p_assigned $(p,r,s) \rightarrow$ can_be_acqured(p,r,s)
4. can_activate$(u,r,s) \wedge$ can_be_acquired$(p,r,s) \rightarrow$ can_acquire(u,p,s)

Fig.1 illustrates the basic operations of group situation based cooperation model. Users are mapped to roles based on capability matching and QoS parameters filtering as well as context-based constraints. Users mapped roles are assigned proper permissions to cooperate with each other depending on changes of group situations.

5 Illustrative Examples

In this Section, we revisit the scenarios depicted in Section 2 to illustrate the proposed model. Fig. 2 shows role description for the example.

Role *presenter*
capability: present();
context-constraints : age>30, position= departmentManager;
QoS: trustiness>90, reliability>90; availability=100;
Role *dataAnalyzer*
capability: useDatabase();
context-constraints: part = data-analysis part, availableTime > 5;
QoS: trustiness>80, availability=70;
Role *statistician*
capability: takeStatistics(), useStatisticsProgram(), useDatabase();
context-constraints : part=statistics, availableTime>7, position > sectionChief;
QoS: trustiness>90, reliability>80; availability=80;
Role *sectionResearcher*
capability: none;
context-constraints : availableTime>5;
QoS: trustiness>60, availability=80;
Role *documentWriter*
capability: useDatabase(), makePresentationFile(), useDrawingProgram()
context- constraints: position > sectionChief, availableTime>7
QoS: trustiness>90, reliability>90; availability=90;

Fig. 2. Role Description of '*preparing presentation*' example

Five roles are designed for achieving the common goal, and four group situations are defined to cooperation among roles. In the starting situation s_0, all roles are mapped to roles, and permissions are assigned to roles based on group situation. As we can see in cooperation description, a group situation is expressed by user context.

Table 2. Elements in '*preparing presentation*' example

USERS	$\{u_1,\ldots,u_6 \mid$ John, Daniel, James, David, Jack, Joe$\}$
ROLES	$\{r_1,\ldots,r_5 \mid$ Presenter, Data analyzer, statistician, section researcher, document writer $\}$
OBJS	$\{o_1,\ldots,o_6 \mid$ employ-database, presentation-file, statistic-file, data-analyzing-file, research-section-file, presentation-room$\}$
OPS	$\{op_1, \ldots, op_5 \mid$read, write, modify, delete, enter$\}$
PERS	$\{p_1,\ldots,p_{13} \mid (op_1,o_1), (op_2,o_4), (op_2, o_5), (op_1,o_4), (op_3,o_4), (op_1,o_5), (op_2,o_3), (op_3,o_5), (op_2, o_2), (op_1, o_2), (op_3, o_2), (op_5, o_6),(op_3,o_4)\}$
SITUATIONS	$\{s_0,s_1,s_2,s_g \mid$Starting, TakingStatistics, MakingPresentationFile, Presentation$\}$
CAPS	$\{c_1,\ldots,c_6 \mid$present(), useDatabase(), takeStatistics(), useStatisticsProgram(), makePresentationFile(), useDrawingProgram()$\}$
CONTETS	$\{x_1,\ldots,x_4 \mid$age,position,part,time$\}$
QOSS	$\{q_1,\ldots,q_3 \mid$trustworthyness, reliability, availability$\}$

Users mapped to roles share objects within the group, and permissions for those objects may change with time. For example, presentation file is authorized to Joe (r_4) for writing operation in s_2, and to John and Joe for reading operation in s_g. This dynamic permission assignment supports the success of cooperation - each user can act or interact without any limitation for accessing the objects or without conflicts with other's accesses. Table 2 lists elements shown in the example.

Predicate-expressions are listed by separated situation in Table. 3. Second column shows run-time events indicated by axioms described in chapter 4.

Table 3. Predicate expression of example 'preparing presentation'

Situation	Assignment(mapping)	Run-time Event
s_0	u_mapped (u_1, r_1, c_1, (x_1,x_2), (q_1,q_2,q_3), s_0) u_mapped (u_2, r_2, c_2, (x_3,x_4), (q_1, q_3), s_0) u_mapped (u_3, r_2, c_2, (x_3,x_4), (q_1, q_3), s_0) u_mapped (u_4, r_3, (c_2,c_3,c_4), (x_2,x_3,x_4), (q_1,q_2,q_3), s_0) u_mapped (u_5, r_4, Ø,x_4, (q_1, q_3), s_0) u_mapped (u_6, r_5, (c_2,c_5,c_6), (x_2,x_4), (q_1,q_2,q_3), s_0)	can_activate(u_1,r_1,s_o) can_activate(u_2,r_2,s_o) can_activate(u_3,r_2,s_o) can_activate(u_4,r_3,s_o) can_activate(u_5,r_4,s_o) can_activate(u_6,r_5,s_o)
	p_assigned(p_1,r_2,s_0) p_assigned(p_2,r_2,s_0) p_assigned(p_3,r_4,s_0)	can_acquire(u_2,p_1,s_0) can_acquire(u_3,p_1,s_0) can_acquire(u_3,p_2,s_0) can_acquire(u_3,p_2,s_0) can_acquire(u_5,p_3,s_0)

Table 3. (*continued*)

Situation	Assignment(mapping)	Run-time Event
s_1	p_assigned(p_4,r_2,s_1) p_assigned(p_5,r_2,s_1) p_assigned(p_1,r_3,s_1) p_assigned(p_4,r_3,s_1) p_assigned(p_6,r_3,s_1) p_assigned(p_7,r_3,s_1) p_assigned(p_6,r_4,s_1) p_assigned(p_8,r_4,s_1) p_assigned(p_9,r_5,s_1)	can_acquire(u_2,p_4,s_1) can_acquire(u_3,p_4,s_1) can_acquire(u_2,p_5,s_1) can_acquire(u_3,p_4,s_1) can_acquire(u_4,p_1,s_1) can_acquire(u_4,p_6,s_1) can_acquire(u_4,p_7,s_1) can_acquire(u_5,p_6,s_1) can_acquire(u_5,p_8,s_1) can_acquire(u_6,p_9,s_1)
s_2	p_assigned(p_{10},r_1,s_2) p_assigned(p_{11},r_1,s_2) p_assigned(p_{12},r_2,s_2) p_assigned(p_{13},r_3,s_2) p_assigned(p_{12},r_4,s_2) p_assigned(p_{11},r_5,s_2)	can_acquire(u_1,p_{10},s_2) can_acquire(u_1,p_{11},s_2) can_acquire(u_2,p_{12},s_2) can_acquire(u_3,p_{12},s_2) can_acquire(u_4,p_{13},s_2) can_acquire(u_5,p_{12},s_2) can_acquire(u_6,p_{11},s_2)
s_g	p_assigned(p_{10},r_1,s_g) p_assigned(p_{10},r_5,s_g)	can_acquire(u_1,p_{10},s_g) can_acquire(u_6,p_{10},s_g)

6 Conclusions and Future Works

In this paper, we have addressed the issues of access control in cooperation systems. We focused mainly on group situation beyond individual context as a key aspect of cooperation systems. Being aware of the group situation is an emerging requirement for cooperative systems such as MAS. Since objects are shared within the group, there might be conflicts or limitation of accesses to resources during cooperation. For achieving the goal of a cooperative group successfully, permissions which are required during cooperation should be given to users within the group. In the proposed model users cooperate with others based on group situation, moreover permission assignment is also based on group situation. Each user in the group acts or interacts with others on a given situation (cooperation), and permissions are given to users on same situation for supporting users' actions (permission assignment). We showed a feasible example based on proposed model with status predicates.

We plan to extend the proposed model in several directions. First, role-hierarchy introduced some RBAC models can be considered for efficient role and permission management. Moreover, for cooperation among users hierarchical relationship can be helpful. Second, we plan to extend the policy specification - language to conform with web-standard.

Acknowledgments. This research is supported by Foundation of ubiquitous computing and networking project (UCN) Project, the Ministry of Knowledge Economy (MKE) 21st Century Frontier R&D Program in Korea and a result of subproject UCN 08B3-S2-10M and this research also supported by the US National Science Foundation award IIS-0545912.

References

1. Wooldridge, M., Jennings, N.R.: The Gaia Methodology for Agent-oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3, 285–312 (2000)
2. Jung, Y., Lee, J., Kim, M.: Community Computing Model supporting Community Situation based Cooperation and Conflict Resolution. In: Obermaisser, R., Nah, Y., Puschner, P., Rammig, F.J. (eds.) SEUS 2007. LNCS, vol. 4761, pp. 47–56. Springer, Heidelberg (2007)
3. Cabri, G., Leonardi, L., Zambonelli, F.: BRAIN: A Framework for Flexible Role-based Interactions in Multi-agent Systems. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 145–161. Springer, Heidelberg (2003)
4. Kim, M., Jung, Y., Lee, J., Kim, M.: Context-based Cooperation Architecture for Ubiquitous Environment. In: Youn, H.Y., Kim, M., Morikawa, H. (eds.) UCS 2006. LNCS, vol. 4239, pp. 171–182. Springer, Heidelberg (2006)
5. Kim, M., Lee, J., Kim, M.: Group Situation based Cooperation Model. In: 2nd International Conference on Convergence Information Technology, pp. 1372–1377. IEEE Computer Society Press, Los Alamitos (2007)
6. McCarthy, J., Hayes, P.J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence 4, 463–502 (1969)
7. Reiter, R.: The situation Calculus Ontology. Electronic News Journal on Reasoning about Actions and Changes (1997), http://www.ida.liu.se/ext/etai/rac/notes/1997/09/note.html
8. Yau, S.S., Wang, Y., Karim, F.: Development of Situation-Aware Application Software for Ubiquitous Computing Environments. In: 26th Annual International Computer Software and Applications Conference, pp. 233–238. IEEE Computer Society Press, Los Alamitos (2002)
9. Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing, Special Issue on Situated Interaction and Ubiquitous Computing 5(1), 4–7 (2001)
10. Sandhu, R.S., Coyne, E.J., Feinsteing, H.L., Younman, C.E.: Role-Based Access Control Models. IEEE computer 29(2), 38–47 (1996)
11. Yuan, E., Tong, J.: Attributed based Access Control (ABAC) for web services. In: IEEE International Conference on Web Services, pp. 561–569. IEEE Computer Society Press, Los Alamitos (2005)
12. Priebe, T., Dobmeier, W., Kamprath, N.: Supporting Attribute-based Access Control with Ontologies. In: 1st International Conference on Availability, Reliability and Security, pp. 465–472. IEEE Computer Society Press, Los Alamitos (2006)
13. Kern, A., Walhorn, C.: Rule Support for Role-Based Access Control. In: 10th ACM Symposium on Access control models and Technologies, pp. 130–138. ACM, New York (2005)
14. Kulkarni, D., Tripathi, A.: Context-Aware Role-based Access Control in Pervasive Computing Systems. In: 13th ACM Symposium on Access control models and Technologies, pp. 113–122. ACM, New York (2008)
15. Kuamr, A., Karnik, N., Chafle, G.: Context Sensitivity in Role-Based Access Controls. ACM SIGOPS Operating Systems Review 36(3), 53–66 (2002)
16. Yau, S.S., Yao, Y., Banga, V.: Situation-Aware Access Control for Service-Oriented Autonomous Decentralized Systems. In: International Symposium on Autonomous Decentralized Systems, pp. 17–24. IEEE Computer Society Press, Los Alamitos (2005)

17. Maximilien, E.M., Singh, M.P.: A Framework and Ontology for Dynamic Web Services Selection. IEEE Internet Computing 8(5), 84–93 (2004)
18. Barwise, J., John, P.: Situations and Attitudes. Journal of Philosophy 77, 668–691 (1981)
19. You, S.S., Huang, D., Gong, H., Seth, S.: Development and runtime support for situation-aware application software in ubiquitous computing environment. In: 28th International Conference on Computer Software and Applications, pp. 452–472. IEEE Computer Society Press, Los Alamitos (2004)
20. Joshi, B.D.J., Bertino, E., Ghafoor, A.: Temporal Hierarchies and Inheritance Semantics for GTRBAC. In: 7th ACM Symposium on Access control models and Technologies, pp. 74–83. ACM, New York (2002)