DySCon: Dynamic Sharing Control for Distributed Team Collaboration in Networked Enterprises

Ahmad Kamran Malik, Hong-Linh Truong, Schahram Dustdar
Distributed Systems Group
Vienna University of Technology, Austria
{kamran, truong, dustdar}@infosys.tuwien.ac.at

Abstract – **Networked enterprises create virtual teams of distributed experts belonging to different enterprises where one user can be part of multiple teams; How to effectively control the sharing of personal and shared context information among members of multiple overlapping teams without compromising their privacy is a challenging research question. This paper describes sharing control in Peer to Peer and Web service based collaborative systems. In contrast to other sharing systems which mostly use static policy and context of requester and owner, we propose DySCon, a context based Dynamic Sharing Control mechanism which allows system defined as well as owner defined runtime policy adaptation for different levels using various contexts. We evaluate our Dynamic Sharing Control architecture by implementing a prototype Dynamic Sharing Control Messenger to enhance privacy of the owner.**

Keywords-Sharing control, enterprise collaboration, context, dynamic policy.

I. INTRODUCTION

In a dynamic collaboration environment created by networked enterprises [1], one of the important requirements is to decide when to use which context service at what granularity level, when to share with a particular team member to improve the efficiency of collaborative task without compromising the privacy of owner, her team, and the enterprise. From a user's point of view, two types of context services are Personal Context Services, for example, managing and providing user's location and her personal activity, and Team/Enterprise Context Services, for example, managing activities of a team. In this paper we are mainly concerned with issues related to privacy of personal context information. Users need to share their context information with members of different teams which means that private information, like current location or personal activity information, may be handed over to others. As the sharing requirements change with the context, for example location, activity, role or team of the user changes, it is required to dynamically change the enterprise defined sharing policy for a user.

Many context based access control systems for collaborations have been proposed in literature as described in [2]. Most of these systems use the context of requester and provider for sharing context information. Existing systems, as described in Section IV, mostly use static sharing policy defined by the enterprise or team. Some of the recent systems [3] use owner-defined roles instead of the enterprise defined roles like RBAC [4], which can create role management and scalability issues. To support the privacy of user's context info in dynamic collaborations, we present Dynamic Sharing Control architecture (DySCon) which makes use of context of the requester, provider, resources, activities, teams and enterprises. These contexts are used by DySCon for sharing control at three levels: i) empowering enterprise defined roles by context constraints to automatically adapt policy based on context, ii) allowing owner to modify her sharing policy at runtime using predefined templates like busy, not at work etc, and iii) allowing owner to modify policy for one or more users, activities, teams and enterprises.

The remainder of the paper is organized as follows. Section II describes a motivating scenario. Section III describes context sharing and control. Section IV gives the background and related work. Section V shows the DySCon architecture. Section VI contains dynamic sharing control policy. Section VII describes the implementation and discussion of our architecture. Section VIII concludes the paper and outlines the future work.

II. MOTIVATING SCENARIO

We describe a scenario in service-oriented collaborative systems (e.g., those developed in [5]) where users[1], teams, enterprises, activities, and services are involved entities. Networked enterprises create virtual teams whose members are selected from different enterprises based on their skills and experiences. Teams are dynamically created and dissolved, team members are distributed and dynamic which join and leave when required. Each team is assigned to one or more activities where activities can be divided into sub-activities. Services are used in activities which are being performed by team members. Context of the user environment and her personal activities is stored at user's peer which can be accessed by other users calling her personal context services. We are interested in dynamic sharing control of the personal context services. Users share these services with others to allow their context to be accessed.

This work is partially supported by the European Union through the FP7-216256 project COIN.

[1]User is also referred to as member, owner, and subject in this paper

978-0-7695-3755-9/09 $25.00 © 2009 IEEE
DOI 10.1109/CEC.2009.55

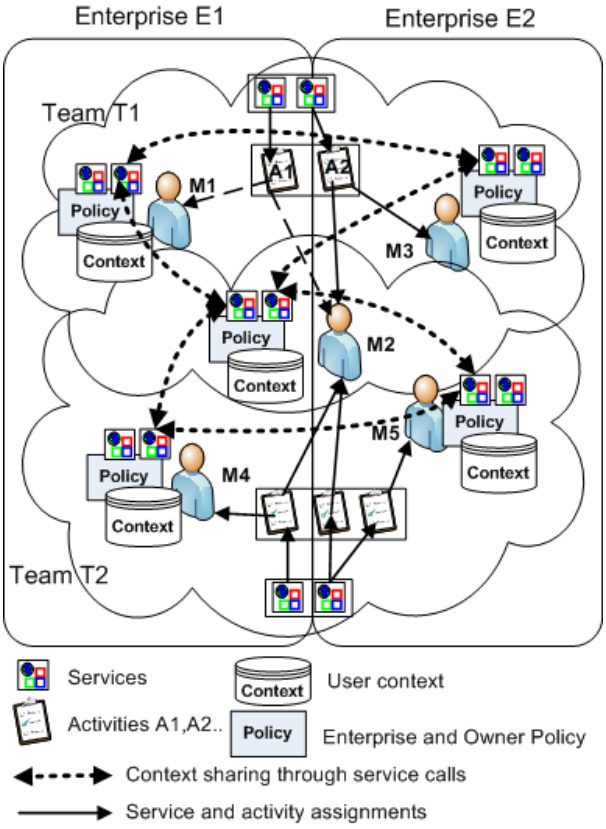

Fig. 1: Motivating Scenario

Consider an example of virtual team collaboration for the above scenario as shown in Figure 1. M1 and M2 are members of virtual team T1 which is mutually created by two enterprises E1 and E2. M1 and M2 are employed in E1 and E2 respectively. M2 is also a member of another team T2. Members of the team T1 can access services from each other controlled by the policy defined for their roles by their enterprises. So M1 and M2 can share each others location and activities at a higher granularity level, for example, location service shares only the city name instead of complete address. M1 can access activity service of M2, but M1 can only see the activity names related to team T1 and see nothing for team T2 activities because she is not member of T2. When T1 assigns an activity A1 (shown by dashed lines) to both M1 and M2, they can share the details of their mutual activity A1 as long as the activity is continued. They can share their location during the office hours because the system automatically allows them by checking their mutual activities, activity context, temporal context and location context. This dynamic collaboration scenario shows the importance of ensuring the privacy of user's context information.

III. CONTEXT AND SHARING CONTROL

The importance of context for sharing control is described in existing systems [6] [3]. Context plays an important role in collaborative systems [7] where activity based situations are changing from time to time. Context helps in capturing such situations and enables the collaborative systems incorporate context conditions to enhance their efficiency.

Context has been defined by Dey in [8] as:

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

This definition reflects the importance of context in different situations and related entities. For our work the importance of context is twofold, first we use the context to share collaborative information, and second we use context to control the sharing of context between entities.

A. Autonomous Peers and Web Services

The distributed, autonomous, and dynamic nature of peer to peer systems [9] make them ideal to be used in our scenario. For interoperability between peer applications, Web services can be the suitable choice with their network-, operating system- and platform-independent characteristics. We implement DySCon as a peer to peer and service-oriented system where peers are autonomous distributed nodes over the Internet which can define own sharing policy using SOAP based Web services to control the sharing of context with other peers.

B. Context Granularity and Sharing Control levels

In a dynamic collaboration scenario, where users from different enterprises are working for more than one teams in various context conditions, it is not possible to share same level of context information with each of them. For example, a user would like to share details of her activity context with the users who are part of same activity, and only high level of activity context with users who are members same team but not participating in this activity, and nothing with members of the other teams. In DySCon we manage hierarchical levels of context granularity for each context item as described in [10]. For example the location context can be managed in three hierarchy levels as city name, street address and room level address.

A user may wish to restrict at entity level. For example a user may wish to restrict another single user or all the members participating in an activity, team, or enterprise. For the privacy of the owner context, she is able to control sharing with other users at all entity levels. In DySCon we implement these restrictions at service level using the context granularity.

C. Dynamic evaluation of context based request

In our system context serves multiple purposes; sharing context information, control of sharing context and dynamic adaptation of sharing policy based on context. Context sharing control is performed in two steps. First the user sends a request stating her role and the context service she wants to access. The owner system sends a reply which consists

of the required context of the requester for context access. If the requester wishes to share her context then in second step the requester sends the complete request including her current context requested. The evaluation is performed at the owner's system and if all context conditions are fulfilled, required context is shared with requester. The second step can be manual or automatic depending on requester's choice, so that the requester's system can automatically send all the required context information.

To allow the owner to preserve her privacy in sharing, owner-based rules override the enterprise defined policy. Still these rules are defined by owner in certain conditions and are mostly used temporarily whereas the enterprise policy is rather static and works in all cases when an overriding rule does not exist.

IV. BACKGROUND AND RELATED WORK

In collaborative working environments (CWE) [5], privacy of the user context is considered as a fundamental requirement. In the past, sharing control used a simple two dimensional matrix, which was modeled as access control list or capability list to describe who can have what level of access to which objects [11]. These lists are not scalable as access right management is difficult with large number of users and rights. In [12], subject object based access control model is presented for collaborative environments where it identifies different roles of users and their collaborative rights. The Role-Based Access Control model is a natural way to define groups of users and rights. RBAC models are described in [4]. RBAC is a model to group number of rights into a role which can be granted to one or more users. User can activate a role within a session. This idea makes management of rights very easy and makes it a scalable model. It reduces cost and complexity of the security administration. The importance of creating groups for managing access rights is visible from the awareness study [13]. The use of RBAC in collaborative systems is described in [14]. A survey is presented in [2] about access control systems for collaborative environments and their requirements. In DySCon, we extend RBAC to incorporate context constraints.

Some current systems have made use of context in RBAC such as [6], [15], [16]. Context can help in defining context constraints for fine grained access control and runtime adaptation of access control policy [17]. Policy adaptation is performed using semantic technologies e.g., [18] to specify context and policy for high level context description and reasoning.

The RBAC model has been extended to support for collaborations. The notion of team is used in Team-Based Access Control models [19]. Access control has also been proposed to support task activities as in Task-Based Access Control model [20]. At each step of the task there are certain rights.

Access control has also been used to access Web services using role and context in [21], [22] and [23], but these systems are not used for ensuring privacy to individual context. In [3] user-defined roles have been used to provide privacy to individuals. Using only owner defined roles creates role management and scalability problems.

We observed that most of the systems are static, a few recent systems use context based policies mostly using requester and owner context. Our system provides active access control using context of all entities and uses context for context access. Fine grained level of access is provided in the presence of role based access. In addition we use service based and peer to peer technologies.

V. DySCon ARCHITECTURE

The Dynamic Sharing Control (DySCon) architecture is shown in the Figure 2. We use a peer to peer and Web services based middleware system described in [24]. In our system each team member is represented by a peer and each team is also a peer which is controlled by a team leader. The team peer tells all its members about other members of the team. Main components of the architecture and their interconnections which are described in the following paragraphs.

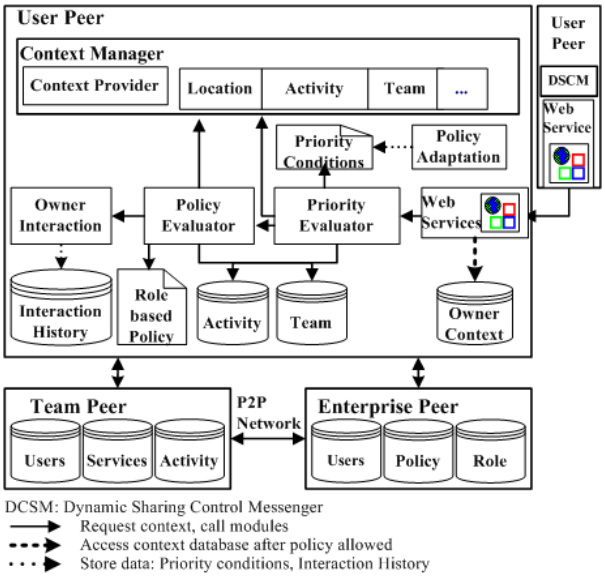

Fig. 2: DySCon Architecture

A. Policy Adaptation and Priority Evaluator

The *DSCM* is a GUI component which allows the user to view, add, change and delete the existing priority rules defined by the user himself. It also allows the user to view the enterprise policy which the user is not allowed to change. User-defined rules override the existing enterprise policy. Rules are stored in a *Priority Condition* document. When a request arrives the Web Service after accepting the request sends it to *Priority Evaluator* component which is responsible for checking the existing priority rules for the request before sending it to the *Policy Evaluator*. In case the priority rule is found for that request, the request is not sent to *Policy Evaluator* and the context conditions required in the owner defined priority rules are matched with requester current context which is provided by *Context Manager*. If the conditions match the request is allowed otherwise denied and not sent to *Policy Evaluator* or *Owner Interaction* component.

B. Policy Evaluator

When no priority condition is found for the particular request, query is sent to the *Policy Evaluator* which matches the role of requester and searches for its related policy in Role Based Policy document defined by the enterprise. After finding a policy rule for requested service and requestor, it matches context conditions defined for accessing requested service. If the required context conditions are satisfied the requester is allowed to access the service.

C. Owner interaction and history

If the role of the requester allows for the requested service but context conditions are not satisfied then the system calculates the statistics for previous interactions with this requester peer. All the collaborative interactions for sharing context are stored in an *Interaction History* database. The system shows the statistics of previous interactions to the owner who can see how many times in past this peer has requested which service under which context conditions and how many times interaction was successful. This helps owner to decide if she trusts and wants to allow the request, otherwise the request is denied.

D. Context Manager

Each peer contains our DySCon policy which is implemented on top of the middleware. In this way access control policy is completely decentralized and owner of context on each peer is in full control of her context and access policy. Access policy contains the core part of the policy which is usually unchanged and a situation base containing the context based dynamic rules required for temporarily allowing some role in a given context condition.
Context of both the requester and owner is used in evaluating the access control request. We use following types of context features for our access control policy.

- Local context features are the peer and its environment related features. For example personal features and skills, location, resources etc.
- Activity and team related context features like current activities, activity status, scheduled activities, team members in the activities, calendar of activities and team related features.
- History related features like the previous access history, contact history, reputation.

VI. DYNAMIC SHARING CONTROL POLICY

We use RBAC model for our sharing policy. The dynamic policy will include role as well as context of requester, owner and other related entities like current or mutual activities, services, teams and enterprises. We enhance owner privacy by arranging context data in different levels and deciding access to a specific level of context by using role and current context. In addition, the owner of data is allowed to change her access preferences for specific entity. Owner is able to grant or deny specific level of access to some specific users or teams. Sharing based on context can create predefined restriction. For example, a subject is given access to an object based on her role and current context. After some time her context changes for example due to change in location or time period, so the granted access will be revoked as soon as the context is changed. To allow the access, rules can be dynamically changed based on current context conditions i.e., seeing the other related context entities like the continuation of a mutual activity, history of context access of the requestor.

```
<policy>
 <resource>
   <location service>
     <condition>
       <loc>office</loc>
       <time>9:00 to 17:00</time>
     </condition>
     <access level>
       <teammembership>same team</teammembership>
         <level>L1</level>
       <teammembership>different team</teammembership>
         <level>L2</level>
     </access level>
   </location service>
   <activity service>
     <condition>
       <activity status> continued <activity status>
       <loc>office</loc>
     </condition>
     <access level>
       <activitymembership>same activity</activitymembership>
         <level>L1</level>
       <teammembership>same team</teammembership>
         <level>L2</level>
     </access level>
   </activity service>
 </resource>
</policy>
```

Listing 1: Example of Sharing Control Policy for developer role

```
<PriorityRules>
  <priority>
    <resource>activity service</resource>
    <role>developer</role>
    <team>team1</team>
    <activity>
      <name>planning</name>
      <status>continued</status>
    </activity>
    <action>allow</action>
    <Level>L1</Level>
  </priority>
  <priority>
    <resource>location service</resource>
    <role>leader</role>
    <team>team2</team>
    <activity>
      <name>design</name>
      <status>continued</status>
    </activity>
    <action>allow</action>
    <Level>L3</Level>
  </priority>
</PriorityRules>
```

Listing 2: Example of Owner based priority rules

A. Owner based policy

In our system, roles and policies are assigned to individuals by their parents enterprise for sharing context information. An example of DySCon Policy for Developer role is shown in Listing 1. Here context conditions are defined to restrict

access. Also the team and activity memberships are checked for granting level of access. Owner can override these rules by specifying her owner based *Priority Conditions* as shown in Listing 2. Here the owner wants to allow full access (L1) of current activity to the developer of *team1* when planning activity is continued. Owner can also specify positive (Allow) as well as negative (Deny) access rules for any type of entity using context conditions.

Owner based DySCon policy steps are shown in the Figure 3 and the algorithm for resolving the policy rules is described in Algorithm 1.

Algorithm 1 Dynamic Sharing Control Policy Algorithm

```
[Match query and requester context with Priority Rules]
if found priority rule for requester's identity, role or requested service then
    if found negative access then
        Reply "Service Unavailable"
    else if found positive access then
        if matched requester context with required context conditions then
            find level of context access
            Grant permitted level of requested service
        else
            Reply "Service Unavailable"
        end if
    end if
else
    [Match with Policy Base]
    if found requesters role and requested service and context conditions then
        find level of context access
        set revocation rules
        Grant permitted level of requested service
    else
        Reply "Service Unavailable"
    end if
end if
```

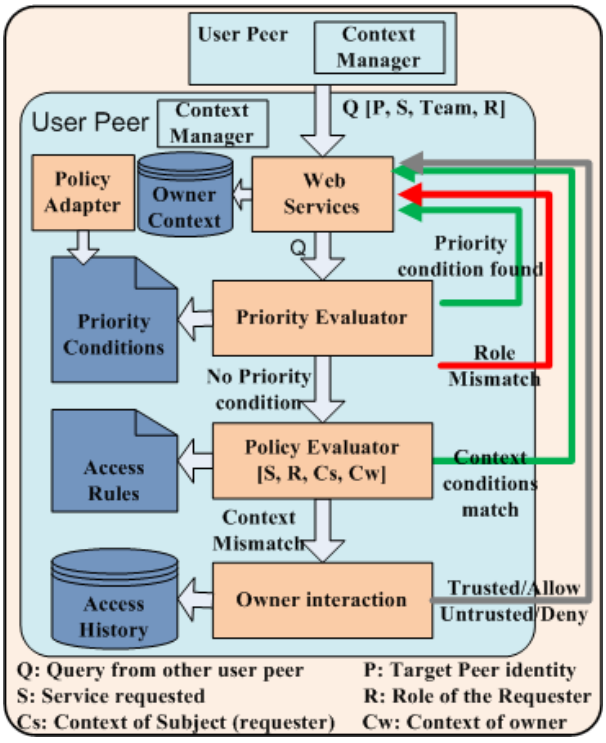

Fig. 3: DySCon Policy Evaluation

VII. IMPLEMENTATION AND DISCUSSION

The prototype of DySCon has been implemented using Web services in Java. We used a peer to peer middleware for Web services invocations, supporting synchronous/asynchronous communication [24]. Context at three level of hierarchies is stored in database within user peer. Context based Enterprise Policy is defined as XML document contains policies for each type of role. Owner based Priority Conditions is also an XML document containing temporary sharing policies.

A. Dynamic Sharing Control Messenger

Figure 4 shows the outlook of our Sharing Control Messenger which provides a GUI to query the online members of the collaborative community. Working of most of the parts of messenger are described by annotations. A user first selects her team after which she is able to see the status of all the members for example online, offline, busy, not at work etc. Selecting an online member will show her current contact details, collaboration details like mutual activities, current status and personal context services for sharing context information.

Services are defined by the system to share the context of collaborating user, for example, location service, activity service, chat service and calendar service etc. By clicking on any of these services a request is sent to the owner of service for sharing context. The other system replies according to her enterprise and owner policy as shown the reply of Current Activity Service in Fig. 4. Most of the time collaborating users in a team get positive reply which may be constrained by the granularity of context, for example, the requester will not be able to see the details of activity or location of other user who is not in a mutual activity with him at this particular time. Still she will get the high level of context result like the city name for location. For calendar service she is able to see only the activities shown in calendar related to their common team, for all other activities like activities of other team or owner's personal activities she gets *not available*. The chat service is accessible only if the other user has enabled chat option for requesting user or her team.

The other part of the system is related to dynamic adaptation of owner policy. As shown in the upper part of messenger, *user policy* area allows the owner to define or change her own policies for a particular level which she can select by clicking the given options for user, role, team, activity, service and enterprise.

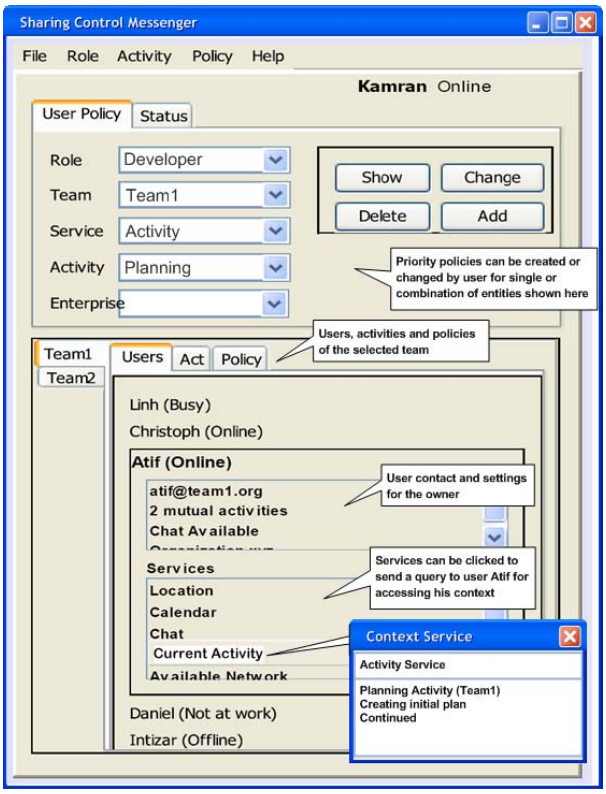

Fig. 4: Dynamic Sharing Control Messenger

The importance of the application for collaborative team environments is visible from this system where owner feel free to share her context with the other team members whom she never met or see and can not trust. This system preserves the owner's privacy by allowing him to control her sharing rules for others at at level of details. The level of granularity comes in two flavors, Context Level Granularity and Entity Level Granularity both work at same time to preserve the owner's privacy to any level of detail.

VIII. CONCLUSION AND FUTURE WORK

In this paper the Dynamic Sharing Control (DySCon) is described for distributed teams and enterprise collaborations. We have shown our DySCon architecture which extends the RBAC model by including context constraints. We also show the use of Priority Condition rules for getting priority over Role based Access rules defined by enterprises. It makes the system context aware and owner based which is very useful for owner's privacy. Context is used at all entity levels and different levels of granularity which is an important requirement of collaborations. Collaborating team members can share context up to a detail level based on their current level of collaboration with the other user. History is used as another factor to make dynamic decision based on previous interactions of the user.

In future we want to make the system more user friendly by creating owner based roles, assign them to other users on request and handling their management and scalability issues as well as the evaluation of the system. Context types, services and their use will be enhanced to handle dynamic scenarios and applications of multiple team collaborations.

REFERENCES

[1] L. Camarinha-Matos and H. Afsarmanesh, *"Collaborative Networks: Reference Modeling"*. Springer Publishing Company, Incorporated, 2008.

[2] W. Tolone, G. J. Ahn, and T. pai, "Access control in collaborative systems," ACM Survey, 2005.

[3] C. Groba, S. Gross, and T. Springer, "Context dependent access control for contextual information," IEEE ARES, 2007.

[4] R. S. Sanadhu, "Role-based access control models," IEEE, 1996.

[5] "European union project coin," http://www.coin-ip.eu.

[6] M. Convington, "Securing context aware applications using environment roles," ACM, VA, 2001.

[7] "European union project incontext," http://www.in-context.eu.

[8] A. K. Dey and G. D. Abowd., "Towards a better understanding of context and context-awareness," Workshop on The What, Who, Where, When, and How of Context-Awareness, (CHI 2000), The Hague, The Netherlands, April 3, 2000.

[9] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, and J. P. *et al.*, "Peer-to-peer computing," HP Laboratories, HP Laboratories Palo Alto, Tech. Rep. HPL-2002-57 (R.1), July 2003.

[10] C. Dorn and S. Dustdar, "Sharing hierarchical context for mobile web services," *Distributed and Parallel Databases*, vol. 21, no. 1, pp. 85–111, February 2007.

[11] R. Sandhu and P. Samarati, "Access control: Principles and practice." IEEE Communications, 1994.

[12] D. P. Shen H., "Access control for collaborative environments," ACM 1992.

[13] S. Patil and J. Lai, "Who gets to know what when: Configuring privacy permissions in an awareness application," CHI 2005 ACM, Portland, USA.

[14] G. Ahn, "Authorization management for role based collaboration," IEEE int. conf. on System, Man and Cybernetic, Washington, 2003.

[15] S.-H. Park, Y.-J. Han, and T.-M. Chung, "Context-role based access control for context-aware application," HPCC 2006, pp. 572580, Springer-Verlag Berlin Heidelberg 2006.

[16] H. J. Ko, D. H. Won, D. R. Shin, H. S. Choo, and U. M. Kim, "A semantic context-aware access control in pervasive environments," Springer-Verlag Berlin Heidelberg 2006, ICCSA 2006, pp. 165 174.

[17] Y.-G. Kim, C.-J. Mon, D. Jeong, J.-O. Lee, C.-Y. Song, and D.-K. Baik, "Context-aware access control mechanism for ubiquitous applications," AWIC 2005, pp. 236242, Springer-Verlag Berlin Heidelberg 2005.

[18] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila, "A semantic context-aware access control framework for secure collaborations in pervasive computing environments," ISWC 2006, pp. 473 486, Springer-Verlag Berlin Heidelberg 2006.

[19] R. Thomas, "Team-based authorization control(tmac)," ACM, VA, 1997.

[20] R. Thomas and R. Sandhu, "Task-based authorization controls(tbac): Models for active and enterprize-oriented management," Database Security XI, Holland, 1997.

[21] S. Haibo and H. Fan, "A context-aware role-based access control model for web services," IEEE International Conference on e-Business Engineering (ICEBE05).

[22] V. Kapsalis, L. Hadellis, D. Karelis, and S. Koubias, "A dynamic context-aware access control architecture for e-services," computers and security 25(2006)507521.

[23] M. Coetzee and J. Eloff, "A trust and context aware access control model for web services conversations," TrustBus 2007, pp. 115124, Springer-Verlag Berlin Heidelberg 2007.

[24] L. Juszczyk and S. Dustdar, "A middleware for service-oriented communication in mobile disaster response environments," 6th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC)2008, Leuven, Belgium.