CONTEXT-AWARE SHARING CONTROL USING HYBRID ROLES IN INTER-ENTERPRISE COLLABORATION*

Ahmad Kamran Malik and Schahram Dustdar
Distributed Systems Group, Institute of Information Systems, Vienna University of Technology, Vienna, Austria

Keywords: Role-based access control, Context-aware systems, Context sharing, User-defined roles, Inter-enterprise collaborations.

Abstract: In enterprise-based collaborations, humans working in dynamic overlapping teams controlled by their respective enterprises, share personal context and team related context for accomplishment of their activities. Privacy of personal context becomes vital in this scenario. Personal context contains information that user may not want to share, for example, her current location and current activity. We propose a role-based dynamic sharing control model which is owner centric and extends role-based access control model. We provide privacy of owner's personal context by separating it from team related context through the use of owner defined roles. Owner has full control of her personal data and is able to dynamically change her own access rules facing any new situation. We describe a role-based dynamic sharing control architecture which makes use of enterprise-defined roles as well as owner-defined roles for separating user context from team context. We evaluate our approach by providing a real world scenario, its running example, and implementation as sharing control messenger using Web services in Java.

1 INTRODUCTION

With the advancement of computing and distributed systems technologies, new opportunities and challenges arise in the area of Collaborative Working Environment (CWE), for example, (EU-Project, COIN). Collaborative systems are used to handle collaborative work of users as a team to support their activities. Members of dynamic overlapping teams can join and leave team at any time and can be a member of more than one team at a time. Requirements of these systems are not only related to individual work of a team member but also their collaborative team activities and working environment. Privacy of user context and collaborations information in such team environments is an important concern.

Context plays an important role in dynamic collaborative systems. Using dynamic nature of context, access rule adaptation can be performed at runtime (Baldauf et al., 2007). It is possible to dynamically adapt behavior of a system by capturing current context of context requester, provider, resources, and, environment. In dynamic collaborative systems fine-grained level of sharing control is required in presence of dynamic entities in the system. Systems based on Role-Based Access Control (RBAC) model (Sandhu et al., 1996) have been used for controlling user's access to resources.

Owner should have control of her data and be able to dynamically change her sharing rules facing any new situation. In enterprise-based team environments (Dustdar, 2004), enterprises create roles for their users which define their access control policy. We use owner-defined roles in addition to enterprise-defined roles so that owner is able to share her personal context with other collaborating users according to her personal requirements. Owner-defined roles are used for temporary collaborations and are revoked after the sharing requirement is complete. Role revocation is based on static rules such as time duration and dynamic rules such as an external event, agreement, or collaboration history.

We propose a role-based dynamic sharing control model for our collaborative team based scenario. Our system is owner centric and extends RBAC by using various types of context and owner-defined roles. We present a hybrid role-based system using enterprise-defined roles *E-Role* and owner-defined roles *O-Role*, their life cycle including role assignment, usage, and

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

revocation rules. We describe a role-based dynamic sharing control model, its architecture, and implementation using Java based Web services environment. We evaluate our system using an example and scenario related to sharing control in multiple overlapping teams and enterprises. A role-based dynamic sharing control messenger is presented as a real world application with a running example. These examples show the importance of our scenario and model in real world to preserve user privacy in complex collaborative environments.
The remainder of this paper is organized as follows. Section 2 describes background and related work. section 3 shows motivating scenario and role-based dynamic sharing control model. Section 4 explains role-based dynamic sharing control. Section 5 describes role-based dynamic sharing control architecture. Section 6 provides implementation and discussion. Section 7 concludes the paper and introduces future work.

2 BACKGROUND AND RELATED WORK

In dynamic collaborative environments, context-based sharing control and fine-grained level of sharing control are fundamental requirements. RBAC model has been extended using context, for example, system presented in (Covington et al., 2001) uses context to dynamically change access rights. Context is modeled as context roles in (Park et al., 2006). Our system DySCon (Malik et al., 2009) extends RBAC using context of requester, owner, and environment and presents context-based dynamic sharing control model. It tries to provide sharing control for owner context where owner-defined policy rules override role-based policy of enterprise. In this paper, we extend DySCon to provide owner-defined roles to requesters. Owner-defined roles can be revoked using predefined context conditions. In (Groba et al., 2007), owner sends one or more owner-defined roles to requester who can select one of them according to her requirements. Difficulties with this system are role selection, role creation without knowing the capabilities of user in enterprise. Our system uses two types of roles; conventional enterprise-based roles called *E-Role*, and owner-defined roles called *O-Role*. We define link between two types of roles which is used for selecting owner-defined role for a user based on her enterprise-defined role. We also describe role usage and revocation strategies which help in preserving privacy of owner's context information.

Role-Based Access Control (RBAC) model (Sandhu et al., 1996) has been widely used in collaborative systems due to its scalable nature and ease of maintenance (Tolone et al., 2005). It reduces cost and complexity of access control administration. Access control systems (Shen and Dewan, 1992), (Ahn et al., 2003), and (Thomas, 1997) describe different access control models for collaborative environments with different roles of users and their collaborative rights. It is difficult for RBAC to provide fine-grained level of access control because it is based on role (group of users) and not on individual user. In addition, RBAC alone can not handle dynamic environments whose changes can be easily captured by making use of context information. Some of the current systems (Shen and Hong, 2005), (Kapsalis et al., 2006), and (Coetzee and Eloff, 2007) make use of Web services to share context information. A survey on context-aware Web service systems can be found in (Truong and Dustdar, 2009). Our system makes use of Web services and context information in addition to owner-defined roles. We use a p2p based system to handle dynamic nature of teams and relations between peers.

3 MOTIVATION

In this section we describe our role-based dynamic sharing control scenario and present role-based dynamic sharing control model.

3.1 Role-based Dynamic Sharing Control Scenario

Role-based dynamic sharing control scenario is shown in Figure 1. Enterprises *E1* and *E2* collaborate and create two software development teams *T1* and *T2*. Users from enterprises participate in both teams. Some users can be member of both teams at a time, for example, user *U2*. Enterprise-defined roles are already assigned to users, for example, users with developer role are assigned *E-Developer* role, and team leaders are assigned *E-Leader* role. A collaborative activity *A* is created by both teams, users *U1* and *U2* are performing this activity. We describe a scenario where user *U1* is requested by other three users for some services. Each of the three requesters is having different activity and team relationship with user *U1*. For example, user *U2* is involved in the same activity with user *U1*. User *U1* feels no hesitation in assigning her personal context to user *U2* whom she already knows and is collaborating with her. As personal context can only be assigned by owner-defined roles, so user *U1* assigns role *O-Developer* to user *U2*

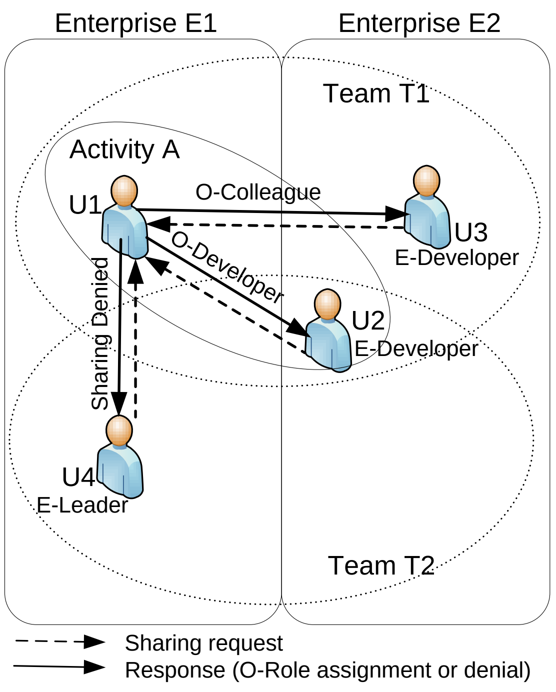


Figure 1: Role-based Dynamic Sharing Control Scenario.

because she is already having *E-Developer* role by enterprise. A revocation rule based on their mutual activity is set automatically by the system which revokes *O-Developer* role from user *U2* as soon as activity *A* is finished. User *U3*, for example, is a trusted colleague of user *U1* in team *T1* so she gets *O-Colleague* role and can share personal context of user *U1* to a level of detail defined for colleagues by user *U1* in her sharing rules for *O-Colleague* role. For colleagues, revocation rule can be set by owner of context *U1* herself, otherwise an event can revoke it, for example, any change in role, team, enterprise, collaboration history of any one of the requester or owner. User *U4* having a powerful role as leader of team *T2* is not able to access personal service of user *U1* because user *U1*'s system does not recognize her as a colleague, friend etc. Her request is logged into the system and if user *U1* herself wants to assign her some role, she can manually assign it.

3.2 Role-based Dynamic Sharing Control Model

Our role-based dynamic sharing control system is based on a dynamic collaborative environment where number of entities are involved, for example, enterprise, team, activity, role, user, and services. These entities described in our scenario are modeled with their interactions, collaborations, and relationships as shown in Figure 2. We explain each these entities below.

Enterprise. Enterprise is the largest entity that creates team, assigns users to teams, and creates roles for users. Enterprise monitors teams and their activities. Two or more enterprises collaborate and create dynamic collaborative environment.

Team. One or more enterprises can create a team and manage it for a joint project. A team is headed by a team leader. A user can be part of more than one teams at a time.

Activity. Activities can be created by enterprise or team. Team leader selects appropriate users for performing activities and monitors their progress. A team can be considered as collection of sub-teams based on groups of users performing activities.

Role. Roles encapsulates rights that are assigned to users based on their job, qualification, and experience. One or more users can have same role and many roles can be assigned to one user.

User. A user works for an enterprise in different teams and activities having one or more roles. User can share her context with other collaborating users and defines her own sharing control rules.

Service. Services in our system are a way to share context and information among users. Personal services are used to share private information of user and shared services are used for sharing activity and team information.

4 ROLE-BASED DYNAMIC SHARING CONTROL

In this section, we describe our hybrid role management scheme and life cycle of an owner-defined role including role assignment and revocation scenarios. Types of roles and evaluation of context sharing request are also described in this section.

4.1 Hybrid Role Management

Two types of roles are used in this system for privacy of owner context and desired level of sharing control.

4.1.1 Enterprise-defined Roles

Enterprises define sharing policies for their employees using enterprise-defined roles *E-Role* which contain rules for sharing control. These rules describe who can share certain services under which conditions. These roles are based on RBAC model. We enhance these roles using context constraints which

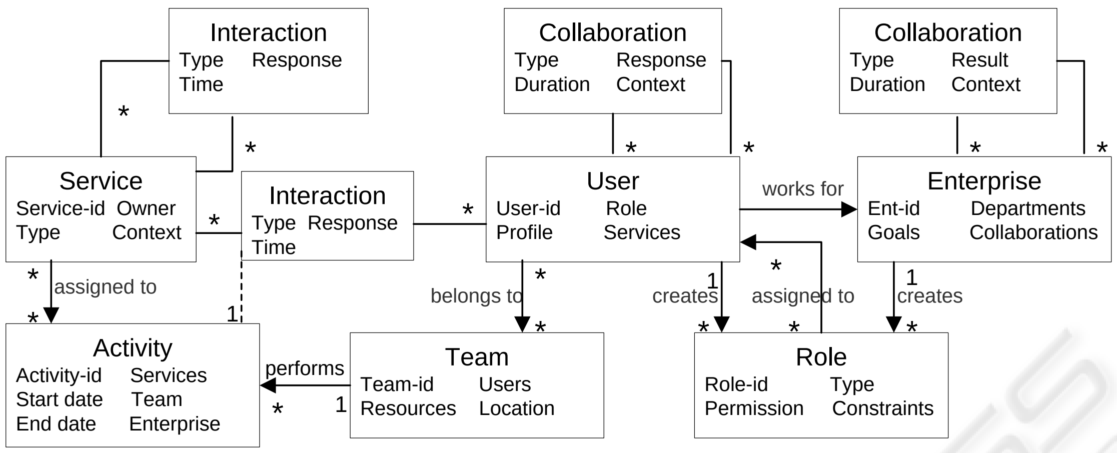


Figure 2: Simplified data model of entities and collaborations.

can also be used for dynamic activation and deactivation of roles.

4.1.2 Owner-defined Roles

Owner-defined roles *O-Role* are used for sharing personal context of owner. These roles are assigned when enterprise-defined role is not able to share context. Assignment of these roles depends on collaboration of owner and requester. There are two types of owner-defined roles which are described below.

***A.* Owner-defined Enterprise-based Roles**

These roles are linked with enterprise-defined roles. If users are participating in a mutual activity, team, or enterprise, system can detect relationships and provide roles related to requester's enterprise-based roles, for example, user having *E-Developer* role will get *O-Developer* role. It provides advantage of personal context sharing at different levels, for example, users having more powerful roles in enterprise will get detailed personal context.

***B.* Owner-defined Personal Roles**

Detailed level of personal context sharing is only available to users like personal friends or family of the owner. Some colleagues from enterprise can also be assigned these roles whom the owner trusts or are her friends. Examples of these roles are *O-Friend*, *O-Colleague*, *O-Family* etc.

4.2 Dynamic Revocation of Owner Roles

Owner-defined roles are provided to friends or few collaborating users to access personal context. These roles must be revoked as soon as they have been used for the said purpose or defined time limit. Their excessive and uncontrolled use can be a threat for user's privacy. Role revocation can be based on static information or dynamic events. Role revocation types are described below.

Role Revocation Scenarios

Following types of static and dynamic rules are used for role revocation.

- Time dependent revocation
  Revoke owner role after a fixed duration of time.
- Event-based revocation
  Revoke role based on some event, for example, activity finished or a change in any one of the user's enterprise, team, activity, or role.
- Context-based revocation
  Revoke role when user context changes, for example, location, online status etc.
- Agreement-based revocation
  Revoke role after some agreement finish or violated. Example of agreement is ”i will share my context if you share same context with me”.
- History-based revocation
  Revoke role after tracking history, for example, no email received from user in last 3 days.

4.3 Handling Context Sharing Request

A high level view of context sharing request using enterprise-defined roles *E-Role* and its response based on owner-define roles *O-Role* including major components of our system is shown in Figure 3.

It describes that E-Role based request is sent to *sharing controller* component of owner system which

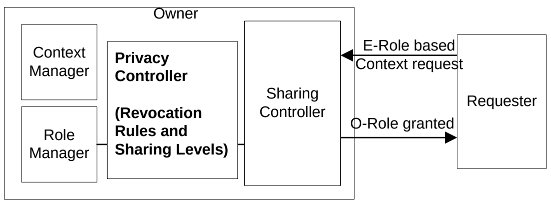


Figure 3: Request for sharing context.

uses other components for request handling. *Privacy controller*, *Context manager*, and *role manager* are used to evaluate request based on sharing rules, current context, and collaboration. To grant a personal context of owner, owner-defined roles *O-Role* are required. *Privacy controller* controls the grant of *O-Role* by defining role revocation rules, and controls context sharing up to certain level of granularity depending on role and context of requester.

5 ROLE-BASED DYNAMIC SHARING CONTROL ARCHITECTURE

Our role-based dynamic sharing control architecture is shown in Figure 4. It is a Web service based peer to peer system where context is shared using Web services. Our architecture consists of *sharing controller*, *privacy controller*, *context provider*, and *role manager*. Other components include *sharing rules*, *collaboration and contact history*, *assigned roles*, and *revocation rules*. Interaction and working of these components is described below.

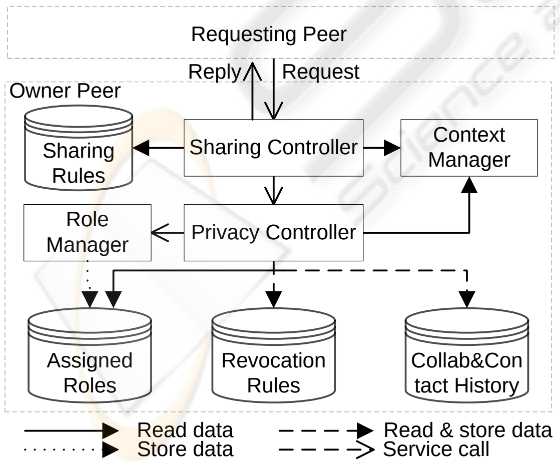


Figure 4: Role-based Dynamic Sharing Control Architecture.

5.1 Sharing Controller

Sharing controller accepts requests from users and provide requested context and roles to user. It uses *sharing rules* to check for validity of user request, for example, requester's role and current context conditions. Sharing control rules are defined by enterprise based on enterprise roles. *Context manager* provides requested context for validating context conditions and for granting requested context. If request is not valid according to sharing control rules, *collaboration and contact history* database is accessed through *privacy controller* to find any collaboration of requester with owner.

5.2 Privacy Controller

Privacy controller uses *collaboration and contact history* database to find requester's collaboration with owner of context. A collaborator is allowed to get owner-defined role *O-Role* to access owner's personal context information. *Role manager* is used for assigning owner-defined roles *O-Role* to requesting users. It also uses static and dynamic revocation rules to revoke a role from user. It can also contact with *context manager* for the user's current context required by revocation rules.

5.3 Context Provider

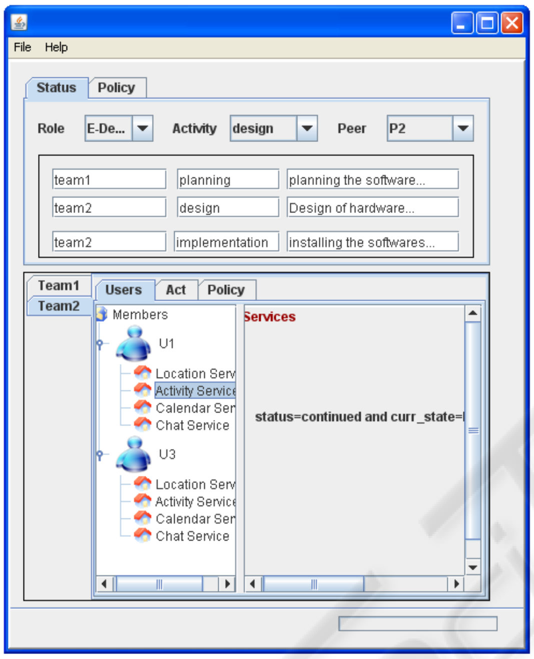
Context provider provides requested context to request handling components. It arranges context information in hierarchical levels, so that only required level of context is shared with a requester. Our system uses context of various types including requester, owner, and all other entities involved in the system. Context is used for context sharing in our system. This means that we use context at two levels, firstly for sharing personal and shared context of users, and secondly for validating sharing rules, revocation rules, and level of sharing based on context conditions.

5.4 Role Manager

Role manager assigns owner-defined roles to requesting users when decided by the *privacy controller*. It provides two types of roles; roles which are defined based on enterprise roles *E-Role*, and roles that are not based on enterprise roles and are used for sharing with friends, family, or trusted colleagues in enterprise.

6 IMPLEMENTATION AND DISCUSSION

Role-based dynamic sharing control system is implemented using Java Web services and is designed as a peer to peer system. Each user, team, and enterprise is a peer. Collaborating users can request context services from other users using Web service calls. We describe the importance of our work in real world scenario presented in Section 3.1 using role-based dynamic sharing control messenger shown in Figure 5.



Role E-De... Activity design Peer P2

team1	planning	planning the software...
team2	design	Design of hardware...
team2	implementation	installing the softwares...

status=continued and curr_state=

Figure 5: Role-based Dynamic Sharing Control Messenger.

As described in our scenario in Section 3.1, users *U1*, *U2*, *U3* and *U4* are collaborating in overlapping teams *T1* and *T2*, created by enterprises *E1* and *E2*. Figure 5 describes one of the user peers owned by user *U2*. The user *U2* can watch her online collaborators who are user *U1* and *U3*. Personal services of each user *U1* and *U3* are available under their names which can be accessed by *U2*. As user *U1* and *U2* are performing same activity *A* which is actually the software *design activity* for developers having *E-Developer* role. When user *U2* requests user *U1* by selecting her *activity service*, the *sharing controller* of user *U1* finds that having role *E-Developer* user *U2* is not allowed the activity service. It hands over request to *privacy controller* which validates user *U2* as a collaborator in a recent ongoing *design activity* and so finds her a candidate for assigning an owner-based role *O-Developer*. It also sets revocation rules for this role to user *U2* assignment and sets level of sharing for this role under given context conditions. In this case revocation rule is based on *design activity* duration. Finally, user *U2* is assigned role *O-Developer* and also gets reply from *sharing controller* of *U1*'s peer containing the requested activity context at certain level of detail allowed to this role as shown in Figure 5. Similarly other users *U3* and *U4* can share context from user *U1* using her provided services. According to our scenario in Section 3.1, user *U3* gets *O-Colleague* role being recognized just as a colleague of user *U1*, so she gets the context being shared at a higher level of granularity. The user *U4* gets a *sharing denied* message having no collaboration with user *U1*.

Sharing control scenario and application described above show the importance of privacy of owner context being shared in complex real world scenarios involving multiple entities and their collaborations. Our system allows sharing of context based on owner-defined roles and context conditions up to a certain level of granularity and for certain time to preserve the privacy of owner context.

7 CONCLUSIONS

Context-based and hybrid role techniques are used to provide sharing control and privacy to owner's context information. RBAC is extended to include context constraints and owner-defined roles. Role revocation for *O-Roles* is provided based on different types of rules like static rules, for example, fixed time limit, or dynamic rules, for example, event-based rules. A Web services based architecture of the system is provided which describes working of all components to provide required level of sharing control. Future work includes the use of autonomic techniques for automatic assignment and revocation of owner-defined roles.

REFERENCES

Ahn, G.-J., Zhang, L., Shin, D., and Chu, B. (2003). Authorization management for role-based collaboration. In *IEEE International Conference on Systems, Man and Cybernetics, 2003.*, volume 5, pages 4128 – 4134.

Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *IJAHUC*, 2(4):263–277.

Coetzee, M. and Eloff, J. H. P. (2007). A trust and context aware access control model for web services conversations. In *TrustBus 2007, Regensburg, Germany*, volume 4657 of *Lecture Notes in Computer Science*, pages 115–124. Springer.

Covington, M. J., Long, W., Srinivasan, S., Dev, A. K., Ahamad, M., and Abowd, G. D. (2001). Securing context-aware applications using environment roles. In *SACMAT '01*, pages 10–20, New York, NY, USA. ACM.

Dustdar, S. (2004). Caramba – A process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Journal of Distributed and Parallel Databases*, 15(1):45–66.

EU-Project (COIN). European union project coin. http://www.coin-ip.eu.

Groba, C., Grob, S., and Springer, T. (2007). Context-dependent access control for contextual information. In *ARES '07*, pages 155–161, Washington, DC, USA. IEEE Computer Society.

Kapsalis, V., Hadellis, L., Karelis, D., and Koubias, S. A. (2006). A dynamic context-aware access control architecture for e-services. *Computers & Security*, 25(7):507–521.

Malik, A. K., Truong, H. L., and Dustdar, S. (2009). DySCon: Dynamic sharing control for distributed team collaboration in networked enterprises. In *IEEE Conference on Commerce and Enterprise Computing, CEC 2009, Vienna, Austria*, pages 279–284.

Park, S.-H., Han, Y.-J., and Chung, T.-M. (2006). Context-role based access control for context-aware application. In *HPCC 2006, Munich, Germany*, volume 4208 of *Lecture Notes in Computer Science*, pages 572–580. Springer.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, 29(2):38–47.

Shen, H. and Dewan, P. (1992). Access control for collaborative environments. In *Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work*, pages 51–58.

Shen, H. and Hong, F. (2005). A context-aware role-based access control model for web services. In *IEEE International Conference on e-Business Engineering (ICEBE 2005), Beijing, China*, pages 220–223.

Thomas, R. (1997). Team-based access control (TMAC). In *Proceedings of the 2nd ACM Workshop on Role-Based Access Control (RBAC-97)*, pages 13–22, New York.

Tolone, W., Ahn, G.-J., Pai, T., and Hong, S.-P. (2005). Access control in collaborative systems. *ACM Computing Surveys*, 37(1):29–41.

Truong, H. L. and Dustdar, S. (2009). A survey on context-aware web service systems. *IJWIS*, 5(1):5–31.

SciTePress
Science and Technology Publications