Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 7 (2012) pp. 60-69

Enhanced Sharing and Privacy in Collaborative Virtual Teams

Ahmad Kamran Malik[1], **Schahram Dustdar**[2]

[1]Institute of Information Technology, Quaid-i-Azam University,
Islamabad, Pakistan
ahmadkamran@qau.edu.pk

[2]Vienna University of Technology, Distributed Systems Group,
Argentinierstrasse 8/184-1, 1040, Vienna, Austria
dustdar@infosys.tuwien.ac.at

Abstract: **Privacy concerns keep users from sharing required information in a collaborative environment. There is a need of privacy preserving methods that can enhance flow of information among collaborating users in dynamic teams without compromising their privacy. We describe a user-defined role-based sharing control model and architecture that uses hybrid roles and hybrid sharing control policy for the owner of information as well as the enterprise. It extends the RBAC model to incorporate context constraints, collaborative relationships, and owner-defined roles. Sharing control request evaluation in presence of hybrid roles and hybrid policy is described. An architecture and its implementation using Web services is described that presents methods for sharing context information among collaborating users of the virtual team.**

Keywords: Access Control; Distributed System; Context-Aware System; Collaborative environment; Virtual Teams

I. Introduction

Web 2.0 [22] and Enterprises 2.0 [16] technologies are being used by the enterprises to create virtual teams of experts. These experts are users who work in distributed and dynamic teams created by one or more enterprises to achieve their common goals. These enterprises and their users are working from different locations connected through social software technologies [17]. Sharing of information among users or groups working to accomplish collaborative tasks is a fundamental requirement. It enables individual users to carry out their tasks efficiently and effectively to contribute to a collective work done more than the sum of individual works. Sharing of information also helps in management of complex distributed tasks and individuals. Information being shared during these collaborations can belong to different types (personal information, activity information, context information, device information) and different privacy categories (public information, private information, semi private information). In distributed and dynamic environments users working in different teams can join and leave a team whenever needed. In such environment, users are reluctant to share their information with other users whom they do not know personally. Role, that is assigned to a user based on user's responsibility [12], is one of the basic credential for sharing information but is not sufficient to identify an individual. Other credentials can identify an individual, yet they cannot develop trust among collaborating users. In this case users share only partial or insufficient information with others.

Different techniques have been used for controlling the information sharing, for example, access control techniques like role-based access control (RBAC) [14], team-based access control [33], context-based access control [26], or organization-based access control [34]. Most of these techniques focus on restricting access to information. We describe user-defined role-based sharing control model for enhanced sharing and privacy, to emphasize the importance of enhanced sharing in presence of privacy constraints, which is key characteristic and requirement of virtual teams collaborations.

In this paper, we describe a model for sharing information among collaborating users, performing activities in one or more teams, created by their enterprises. We intend to enhance the sharing of information among collaborating users. The increase in sharing of activity related information increases the efficiency of collaborative activities and their teams. We extend the Role-based access control model [14] to include owner-defined roles in addition to the traditional enterprise-defined roles. Owner, in our system, is the user who wants to share her information with others. With the increase in interaction among users, the mutual trust increases. Using collaborative relationships and history of previous interactions, an owner can assign the owner-defined role to a collaborating user. Owner-defined role enhances the level of information sharing among users. We describe the assignment and revocation scenarios of owner-defined roles in our system. An architecture of the user-defined role-based sharing control system for enhanced sharing and privacy is presented. Our research efforts include user-defined role-based sharing control model using context of all involved entities and the control of information being shared at fine-grained level of all involved entities. An owner can assign owner-defined role to a trusted requester and can modify sharing

policy for any entity, for example, for a specific user, activity, team, or enterprise. Information is organized in hierarchical order and sharing control system provides information at certain level that is allowed to requester. Remainder of the paper is described as follows. Section 2 describes *information sharing control and mobile-based dynamic collaborations*. Section 3 gives details of *user-defined role-based sharing control model*. Section 4 explains the *user-defined role-based sharing control architecture*. Section 5 explains the *user-defined role-based sharing control policy*. Section 6 shows the *related work*. Section 7 describes *conclusion and future work*.

II. Information Sharing Control and Mobile-based Dynamic Collaborations

In this section, we describe access control, its pros and cons in collaborative and dynamic systems. We also describe the term *sharing control* which is used in our system to emphasize the enhancement of owner-controlled sharing. As distributed and dynamic scenarios require the use of smart devices like mobiles in addition to computers and laptops we describe the use of mobile systems and their limitations for sharing of information.

A. Access Control

There is always a tradeoff between information sharing and privacy of the owner's information. Access control keeps information sharing to a restricted level of user's privacy by defining authorization rules. Static access control rules are not suitable for dynamic environments. Authorization decisions in dynamic environments depend on multiple factors including context of all involved entities, collaborative relationship among entities, history of previous interactions, and roles of the participants.

Access control and privacy policies for users have been centrally administered by the enterprises. Standard access control policies, for example, core RBAC [14] were usually static in nature, that do not take into account the context of subject or object. Currently mobile-based dynamic systems are replacing centralized systems and static access control policies with distributed, peer to peer and, collaborative access control policies, in which users can control sharing of their data. Current research efforts are focussed on owner-defined context-based dynamic policies for enhanced sharing of information. It means the current requirement is to shift control of sharing policies from central administrators to distributed users that will help in achieving distributed, enhanced, and fine-grained level of sharing.

Context-based access control systems [26] are being used for providing required level of access control to users. An owner can decide what level sharing is needed with which users. Access rule adaptation can be performed using the dynamic nature of context at runtime. Likewise, it is possible to dynamically adapt the behavior of a system by capturing the current context of requester, provider, resources, and environment. Context-based systems are helpful in fulfilling the changing access requirements of the owner of information.

B. Sharing Control

In our system, for dynamic and collaborative virtual team environment, we motivate use of the term *sharing control* instead of access control. We coin the term *sharing control* which serves important purposes in sharing of information. *Sharing control* emphasizes the fact that control of information sharing should be distributed among owner and administrator. Also that the owner of information should have full control of sharing her own information with others. In our scenario, we distribute the control of sharing between owner of information and the administrator. An administrator designs access control policy based on enterprise-defined roles while the owner has final authority of sharing her information. An owner can override the administrator-defined policy by creating owner-defined roles and providing authorization policy for them. Owner of information wants control of sharing control policies for her personal information and activities. The owner needs to change policies with change in context and collaborative relationships. An owner may need to control sharing of her information at different levels with different users, for example, restricting some user to access certain information or certain level of information (when same information is available in different levels of detail). In this way, different users can be granted different level of access rights based on their role in enterprise, collaborative relationship with owner, and current context.

C. Mobile-based dynamic collaborations

Mobile devices are the heart of distributed and dynamic systems. In collaborative virtual team-based environments, users are mostly distributed and dynamic. Users use various constrained devices like mobiles, PDA's, and laptops to contact with collaborating users of their team during the movements. On one hand, mobile devices are resource constrained devices. Mobiles have limited battery power, memory, and display. On the other hand, due to advances in mobile technologies, their advantage of anytime and anywhere connectivity is dominating their limitations. During movements mobile system act as a proxy for users and their computers, they can share limited amount of data on user's behalf. Thus mobile devices are very helpful in many applications of collaborative working environments. For example, in disaster scenarios, mobiles are an important and sometimes the only way to connect and share required context of situation.

Due to their resource constraints (limited memory, display, and battery) mobile devices need content adaptation. This requires that the information should be organized in a way so that only certain level of it can be shared using mobile devices. Also the volume of information being sent should be based on context of receiver and receiving device. Only a minimum level of information is shared so that low memory mobile devices can store it with less battery consumption and can display on its small screen.

III. User-defined Role-based Sharing Control Model

In this section, we describe user-defined role-based sharing control model. This heart of this model is user-defined roles

Role Type	*Descriptions*	*Example*
E-Roles	Enterprise-defined roles	E-Manager, E-Developer
O-Role(RBAC)	Owner-defined roles based on RBAC	O-Manager, O-Developer
O-Role(Private)	Owner-defined private roles	Friend, Family

Table 1: Types of roles used in our system

and collaborative relationships among all involved entities. Two types of roles are involved in our system. One is conventional enterprise-defined role as used in RBAC and other is owner-defined role proposed in our system [35]. First we describe enterprise-defined and owner-defined roles, life cycle of an owner-defined role including role assignment and revocation scenarios. Next we describe entities and their collaborative relationships including priorities among entities [23].

A. Hybrid Role creation and management

Our system uses two types of roles. Here we discuss what are these two types of roles, who creates, assigns, and owns them, and how they are used. The two types of roles used in our system are called enterprise-defined roles and owner-defined roles. First type of role the enterprise-defined role is conventional role used in RBAC systems. Second type of role owner-defined role emerged from our requirement to allow an owner of information to control sharing of her personal information. Adding these two types of roles in our sharing control model completes our intention of owner controlled sharing. These two types of roles with their subtypes are described here and shown in Table 1.

1) Enterprise-Defined Roles

These are the conventional roles used in RBAC. An administrator of an enterprises defines these roles according to the responsibilities of employees. These roles are used in our system for controlling sharing policies of the employees of an enterprise who are taking part in some collaborative team activity. We represent enterprise-defined roles as *E-Role* that contain authorization rules for access control. These are rather static roles that describe who can share certain information under which conditions. We use context constraints with these roles to make their dynamic use in our scenario. Context constraints can also be used for dynamic activation and deactivation of roles.

2) Owner-Defined Roles

Owner-defined roles represented as *O-Role* in our system are an alternative to enterprise-defined roles used to allow user the control on sharing of her personal information. These roles are used by an owner when the enterprise-defined role does not allow to share information in a particular scenario while the owner wants to share information with a trusted user. Assignment of these roles depends on many factors including current context, collaborative relationships, and history of interactions. Owner-defined roles are either created from existing enterprise-defined roles or they can be created by the owner. These two types of owner-defined roles are described below.

A. Owner-Defined Enterprise-based Roles

This type of roles are created based on the enterprise-defined roles. These roles are pre-defined in the system by creating an *O-Role* for each *E-Roles*. After creation of these roles, an owner can add the authorizations in these roles to allow users to access more information. These roles are assigned to user whose job role is same as *E-Role* on which it is based, for example, user having *E-Developer* role will get *O-Developer* role. When users are participating in a mutual activity, team, or enterprise, system can automatically detect their collaborative relationships and provide roles related to requester's enterprise-based roles. It provides advantage of enhanced information sharing at different levels, for example, users having *O-Role* will get details of information in contrast to the users having only *E-Role*.

B. Owner-Defined Personal Roles

These roles are optional and are defined for the personal friends or family of the owner. Some colleagues from enterprise can also be assigned these roles whom the owner trusts. Examples of these roles are *O-Friend*, *O-Colleague* or *O-Family*. These roles are assigned personally by the owner to the trusted colleagues, family or friends for sharing their personal information. Authorizations for these roles may be very different and depend solely on personal liking and disliking of the owner.

Owner-defined roles are assigned to collaborating users to enhance the sharing of mutual activities and related context information.

3) Role Assignment

Owner-defined roles can be assigned automatically when the system detects collaborative relationships among owner and requester. It depends on the settings by owner, normally owner would like to share activity information with users who are mutually involved in an activity of the team. To some other friends and trusted colleagues owner can manually assign *O-Role* with certain condition or agreement. 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.

4) Role Revocation

Owner-defined roles are revoked from the requester after they have finished with its intended use or after specific time or event happening. It means role revocation is based on either static or dynamic rules which the system can evaluate and execute automatically. In addition an owner holds the right to revoke her owner-defined role from a user who is no more a trusted user. In the following we describe some possible scenarios for role revocation.

- *Time dependent revocation*
  In this case an owner-defined role is revoked after a fixed duration of time.
- *Event-based revocation*
  An event can be a start or end of a mutual activity or

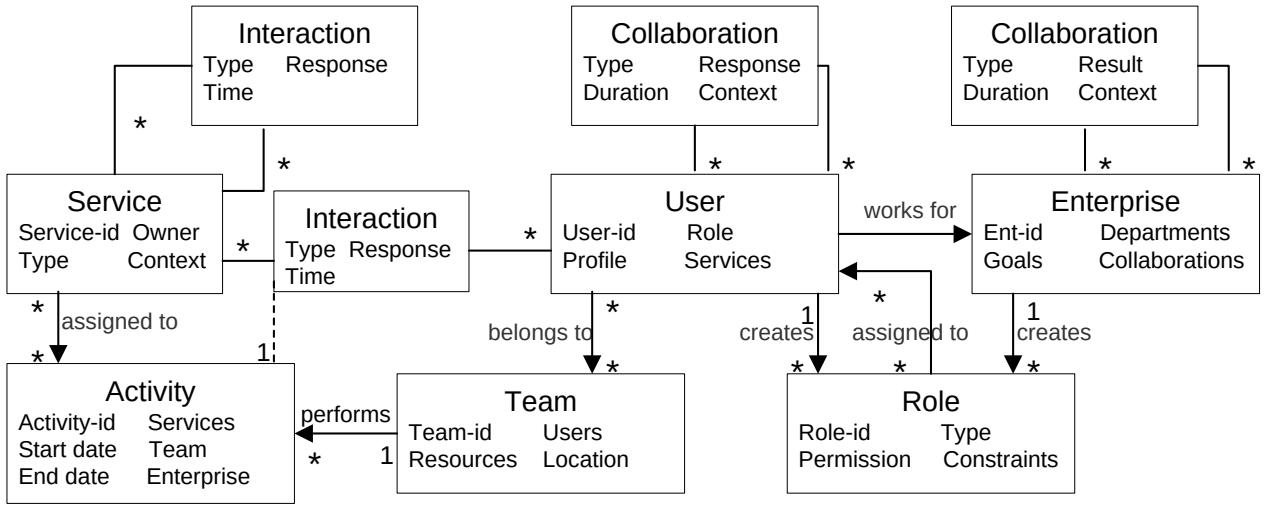

Figure. 1: Simplified data model of entities and collaborations.

a change in a collaborative relationship, for example, change in user's enterprise, team, activity, or *E-Role*.

- *Context-based revocation*
  The *O-Role* assigned to be used only in certain context is revoked when user context changes, for example, a change in location, online status etc.

- *Agreement-based revocation*
  O-Role assigned based on an agreement is revoked after the end of agreement or detection of violation of the agreement. Example of an agreement is "i will share my information as long as you share some specific information with me".

- *History-based revocation*
  Revoke *O-Role* based on history of interactions, for example, no contact from user for the last n days.

B. Entities and Relationships

Our sharing control model is based on hybrid roles as well as entities and their collaborative relationships. There are five entity types used in our system which are enterprise, team, activity, role, and user. Entities in our system are treated differently during sharing control decision because they are given priorities in order of *(enterprise, team, activity, role, and user)* where enterprise is assigned lowest priority and user is assigned highest priority.

1) Entities

Entities described in our system are modeled with their interactions, collaborations, and relationships as shown in Figure 1. Entities involved in this system are described below.

- *Enterprise*
  Enterprise is the largest entity and is assigned lowest priority among entities in our system. Enterprise creates teams, sharing policies (*E-Role*), assigns activities and users to team, and roles to user.

- *Team*
  Team entity can be created by one enterprise independently or by more than one enterprises in collaboration. A team is headed by a team leader who assigns team activities to users. A user can participate in more than one team at a time thus resulting in the concept of overlapping teams.

- *Activity*
  Activities are defined by the team leader and enterprise management. Team leader selects expert users for performing particular activities from within or outside of the enterprise. A team can be considered as collection of sub-teams based on groups of users performing separate activities. Users of these sub-teams can take maximum benefit of our system to enhance sharing among them using *O-Role*.

- *Role*
  E-Roles are assigned by enterprise to users based on their duty, qualification, and experience. *O-roles* are assigned by the automatic system or manually by the owner based on collaborative relationships. One or more users can have same role and more than one roles can be assigned to one user.

- *User*
  A user works for an enterprise in one or more teams and their activities. As an owner of personal and activity information a user can share this information with other collaborating users using *E-Role*. To enhance level of shared information with some trusted and mutually collaborating users she can define her *O-Role* and can change sharing control policy rules for particular users or other entities.

2) Collaborative Relationships

Users working at different entity levels have a collaborative relationship among them. These are relationships among users performing activities in overlapping teams belonging to different enterprises. These collaborative relationships are

described as *Mutual*, *Member*, and *Colleague* in our system. *Member* relationship is described as *Me* while *non-member* is described as *NMe*. Similarly *Mu* and *C* describe *mutual* and *colleague* while *non-mutual* and *non-colleague* are described as *NMu* and *NC* respectively. We describe these three types of collaborative relationships below.

- *Mutual (Mu)*
  A *mutual* relationship between users means that they are involved in same activity. Users in *mutual* relationship share maximum information among them as they have to complete their mutual activity goals within given time limits. These users take the benefit of our *O-Role* to enhance sharing among them.
- *Member (Me)*
  Users working in same team are described having a *member* relationship among them. Members of a team can also benefit from *O-Role* to enhance their sharing within a team.
- *Colleague (C)*
  A *colleague* relationship among users describes that they are working for same enterprise. Colleagues may not be a part of same team and activities.

In practice, there exist complex relationships among collaborating users due to overlapping teams and collaborating enterprises. For example, there can exist a *member* relationship between two users while not being mutual and colleague because they are members of the same team but not employees of the same enterprise and not working in a mutual activity. Due to this complexity of relationships, sharing decisions also become complex and it is difficult to define static *E-Role* authorization policies in such a dynamic and complex environment. It encourages to create fine-grained policies and fine-grained management of information.

IV. User-defined Role-based Sharing Control Architecture

The architecture of our system is shown in Figure 2. In consists of user peers, team peers, and enterprise peers. Web services technology is used to share information among distributed peers. A user can use the services provided by other online users. This architecture of a user system consists of *sharing controller*, *access controller*, and *context manager*. *Sharing controller* evaluates the requests for sharing information. It contains different components for deciding about sharing control policies, collaborative relationships etc. *Access controller* is required only if the owner wants to enhance sharing with another user. It controls role assignment and revocation of *O-Roles*. *Context manager* collects context from sources and manages it in different fine-grained hierarchy levels. Interoperation and working of these components is described below.

A. Sharing Controller

Sharing controller is the main component of architecture that accepts requests from requester and evaluates the request for sharing information. For enhanced sharing it forwards the request to *access controller*. It also contacts *context manager*

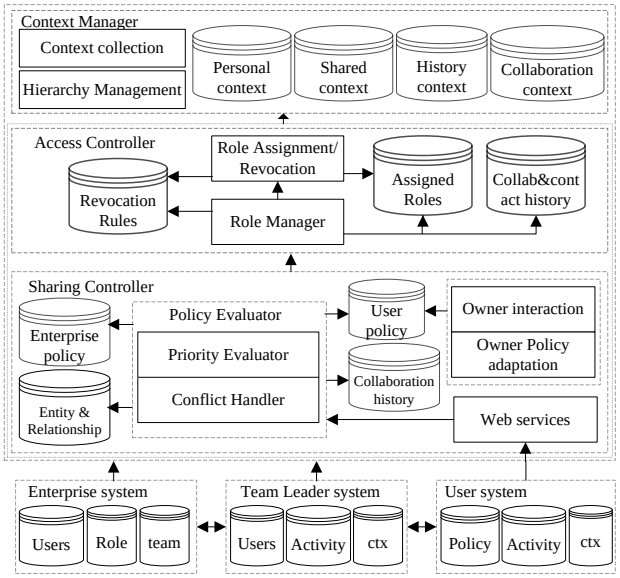

Figure. 2: User-defined Role-based Sharing Control Architecture.

to provide current context for evaluation of the request. It uses *enterprise policy* for *E-Roles* and *user policy* for *O-Role* to check for evaluation of a request. Requester's role and current context conditions are validated first and then it uses *policy evaluator* to check entity relationships, their priorities, and any conflict in policies. After evaluation, either it provides requester with required information at a fine-grained level authorized for this requester or it forwards the request to access controller. Following are the components of *sharing controller*.

1) Enterprise policy and user policy

Sharing control policy in our system is a hybrid policy which is used for defining authorization rules for our hybrid roles. Two types of policies are enterprise-defined sharing control policy and owner-defined sharing control policy (also called user policy). Enterprise-defines RBAC based policy is rather static in nature and we extend it using context constraints in it. Still the enterprise-defined policy is defined by an administrator of enterprise and it is not going to be frequently changed for any entity if required. To allow an owner to share personal information we define owner-defined policy. Owner-defined policy overrides enterprise-defined policy when there is a conflict between policy rules. An owner can change her policy for a user at any entity level when it is required.

2) Policy Evaluator

Policy evaluator uses enterprise policy, user policy, entity relationships, and collaboration history to evaluate the request. It consists of two components *priority evaluator* and *conflict handler*. *Priority evaluator* uses *entity & relationship* to find the collaborative relationship among requester and owner and then decides about the priorities of entities in request and policy rules for that request. First it checks user policy and finds any authorization rule or negative rule for the entity having higher priority in request, for example, if a requester

is *mutually* participating in an activity with owner then any rule about mutual activity gets highest priority. If user policy does not contain any rule about the request then it finds enterprise policy for the request. *Conflict handler* handles policy conflicts mainly using the priorities of entities. If two rules are of same priority then negative rule gets priority over positive authorization rule. Once a policy rule for authorization of request is found context conditions in rule are evaluated.

3) Owner Policy Adaptation

Policy adaptation can be performed only for *user policy*. An owner can change own policy at anytime about a certain user or an entity. In addition system can automatically detect any change in the collaborative relationship, context, or role of a user and adapt the policy for that user at runtime. In this way the *user policy* can be effectively used in distributed and dynamic environments which require frequent changes.

B. Access Controller

Sharing controller forwards the request to *access controller* for providing *O-Role* to collaborating users for enhanced sharing. *Access controller* uses *collaboration and contact history* database to find requester's collaboration with the owner. *Access controller* consists of two components that are *role manager* and *role assignment/revocation*. *Role manager* accepts the request and checks for collaboration and assigned roles and sends request to *role assignment/revocation* component. *Role assignment/revocation* component assigns new role or revokes a previously assigned role.

1) Role Manager

Role manager accepts request from *sharing controller* and finds the collaborative relationship among requester and owner. If requester is a *mutual* in an activity with owner then an *O-Role* based on requester's *E-Role* is assigned. It depends on owner which collaborative relationships gets an *O-Role* and what level of information the requester can access using *O-Role*. User policy defined by the owner actually decides what level of information is accessible to a requester having *O-Role*.

2) Role Assignment/Revocation

Role assignment/revocation component is used for assigning owner-defined roles *O-Role* to requesting users. It uses revocation rules to revoke a role from user. It requests *context manager* for the user's current context to validate revocation rules. It assigns owner-defined roles to requesting users when decided by the *Role Manager* and provides two types of *O-Role* for collaborating partners and for private people.

C. Collaboration & contact history

The history of collaborations among users can be effectively used for enhanced sharing of information. *Sharing controller* and *access controller* both make use of this history. Normally it is used in a scenarios such as the role of a requester allows for the requested information but context conditions or collaborative relationship are not satisfied then the system calculates statistics about previous interactions among requester and owner. When user-defined setting for history are provided, the system can automatically decide what level of recent collaboration history is required to allow for what level of information sharing. On the other hand system provides these interaction statistics to the owner. The owner can check how many times in past this user has requested/-granted which information under which context conditions and how many times interaction was successful or unsuccessful. This build trust in requester and helps owner to decide if she should allow the request.

D. Context Manager

Context manager provides requested context to request evaluating components. Context information is collected from sensors and user's personal information is also provided manually. For example, context is provided by context servers from different sensors in the environment and activity related context is provided by manual entry. For enhanced sharing, context information is arranged in hierarchical levels and only certain level of context is shared with a requester. Context in our system is used for multiple purposes. On one hand it is used to share owner's context information with collaborating partners, on the other hand it is used to evaluate information sharing request. Our system uses context of requester as well as owner to evaluate the information sharing request.

1) Context Hierarchies

All requesters cannot be allowed to access all requested information. It depends on many factors like collaborative relationships, current context, and owner's trust. For example, an owner may want to share more information with users having specific role, context, or relationship. For this purpose our system manages context in fine-grained hierarchies of context information at three different levels named as *L1*, *L2* and *L3*. Level *L1* describes the most detailed information and level *L3* describes the least detail.

2) Types of Context

Following types of context is used in our system.

- Personal context include user, resources, and environment related features. For example, user's location, devices etc.
- Shared context consists of user's activity, team, and enterprise related features, for example, current activities, activity status, scheduled activities, calendar of events.
- History context includes features about history of collaborations with other users, for example, contact history, number of accesses, type of sharing, number and level of successful/unsuccessful sharing.
- Collaborative context, for example, collaborating users in an activity, team, or enterprise, collaborative relationships, collaboration duration.

V. User-defined Role-based Sharing Control Policy

Our sharing control policy includes two types of sharing control rules for a hybrid role-based model. Our system uses Enterprise-defined roles *E-Role* and owner-defined roles *O-Role*. Two types of authorization rules are used for supporting these two types of roles. These two types of rules in our system are described as *enterprise policy* and *user policy*. *Enterprise policy* is based on conventional RBAC roles which are static in nature and cannot be used in dynamic scenarios. We extend RBAC roles by using context constraints. Still RBAC based policy is defined by an administrator in an enterprise and is based on the duty of an employee which cannot be changed easily. For this reason we use *user policy* which is flexible and can be changed according to dynamic requirements of the system.

```
<UserRules>
  <RuleType="regular">
    <Action="+">
      <Subject>
        <Predicate>
          <Op>eq</Op>
          <EntityName> Role </EntityName>
          <EntityFunc> name </EntityFunc>
          <EntityValue>Developer</EntityValue>
        </Predicate>
      </Subject>
      <Object> activity service </Object>
      <Condition>
        <Predicate>
          <Op>eq</Op>
          <EntityName> activity </EntityName>
          <EntityFunc> name </EntityFunc>
          <EntityValue>A1</EntityValue>
        </Predicate>
        <Exp>AND</Exp>
        <Predicate>
          <Op>neq</Op>
          <EntityName> activity </EntityName>
          <EntityFunc> status </EntityFunc>
          <EntityValue> finished </EntityValue>
        </Predicate>
      </Condition>
      <AccessLevel>
          <Predicate>
            <Op>eq<Op>
            <EntityName> Relationship </EntityName>
            <EntityValue> Mu </EntityValue>
          </Predicate>
          <Level> L1 </Level>
      </AccessLevel>
    </Action>
  </RuleType>
</UserRules>
```

Listing 1: An example of enterprise-defined policy

A. Authorization Rules

There are two types of policy rules called regular rules and exceptional rules used in our system. Both of these rules are based on priorities. Regular rules are based on entity priorities that are user, role, activity, team, and enterprise written in highest to lowest priority order. These rules define priority of one entity over another in case of conflict among entities. For example, consider two conflicting rules. One rule says says "sharing of activity information is allowed to users involved in an activity A". Other rule says that "sharing of activity information is not allowed to members of team T". There is conflict in these rules when a user is member of team T and is also involved in activity A. One rule allow sharing

```
<UserRules>
  <RuleType="exceptional">
    <Action="+">
      <Subject>
        </Predicate>
            <Op>eq<Op>
            <EntityName> Team </EntityName>
            <EntityFunc> name </EntityFunc>
            <EntityValue> T2 </EntityValue>
          </Predicate>
      </Subject>
      <Object> activity service </Object>
      <Condition>
        <Predicate>
          <Op>eq<Op>
          <EntityName> Role <EntityName>
          <EntityFunc> type </EntityFunc>
          <EntityValue> O-Role </EntityValue>
        </Predicate>
      </Condition>
      <AccessLevel>
        <Predicate>
          <Op>eq<Op>
          <EntityName> Relationship </EntityName>
          <EntityValue> Me </EntityValue>
        </Predicate>
          <Level> L2 </Level>
      </AccessLevel>
    </Action>
  </RuleType>
</UserRules>
```

Listing 2: An example of user-defined policy

and other rule disallow. This rule is evaluated based on entity priority and is called a regular rule. Smaller entities get higher priority than larger ones because close relationship between users exist in smaller entities. So in this example, entity priority decides that activity has higher priority than team in our system so the user involved in activity is allowed to share information in presence of the fact that members of team T are not allowed. If there is conflict between entities , for example, both rules belong to same entity then negative rule is given priority over positive rule and sharing is disallowed. Our system utilizes the fact that users collaborating in same activity have closer relationships than users involved in different activities. Using these facts our policy supports fine grained level of sharing.

There are some situations when entity-based priority rules are not effective. For example restricting a bigger entity in presence of smaller entity authorizations is ineffective because smaller entity gets priority and will continue to enjoy the sharing. In such situations exceptional priority rules are needed to fulfill user's sharing requirements [8]. Exceptional priority rules are rarely used and are needed in some specific situations and get priority over all other rules.

Our sharing control policy evaluation algorithm is shown in Algorithm 1.

B. Sharing Control Policy

Sharing control rules are explained with example containing regular and exceptional rules as well as positive and negative rules. Listing 1 contains an example of enterprise-defined authorization rules while Listing 2 describes user-defined policy rules. As shown in listing 1 and Listing 2, positive rules symbolized as "+" are used to allow sharing and negative rules symbolized as "-" are used to disallow sharing. Enterprise-defined policy in Listing 1 is mainly based on roles. It shows an example policy for developer role who is allowed to share activity related context information at lev-

el *L1* (most detailed level) from the mutual users in activity named *A1* as long as the activity is not finished. The example of user policy is shown in Listing 2 where user is using an exceptional rule to allow all members of team *T2* to share her activity context at level *L2* (medium level) as long as they hold an *O-Role* provided by the owner.

Algorithm 1 Sharing Control Evaluation Algorithm

```
[Match query and requester context with User Policy]
if found owner − defined 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 if found collaborative relationship then
            Call access controller for enhanced sharing
            Assign O-Role and set revocation rules
            Grant permitted level of requested service
        else if found collaboration history then
            Calculate and evaluate request based on history statistics
        end if
    else if found conflicting rules then
        If exist use exceptional priority rule, otherwise entity priority
    end if
else
    [Match query with Enterprise Policy]
    if found requesters role and requested service and context conditions 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
        else if found conflicting rules then
            use entity priority to evaluate request
        end if
    end if
end if
```

C. Implementation and Discussion

Our user-defined role-based sharing control system is a distributed and dynamic virtual teams based system. It is implemented using Web services in Java and uses peer to peer model. In this system, user, team, and enterprise are the interacting peers. Users can send request for accessing information from other users using Web services provided by other users. We implemented the system as a messenger type application for sharing limited amount of information among collaborating users. Most of the information shared among collaborating users is context information related to their personal or activity information. This limited amount of context information sharing is very helpful in knowing about the status of current activities and their progress among collaborating users and with team leader. Distributed and dynamic virtual teams based systems mostly depend on constrained devices like mobile and PDA's that can share a limited amount of context information among virtual team users.

User-define role-based sharing control model shows the importance of enhanced sharing and privacy of owner context being shared in complex real world scenarios involving multiple entities and their complex collaborative relationships. The sharing control system allows sharing of information based on hybrid roles, collaborative relationships, context conditions, and history of collaborations. It enhances the sharing of information and at the same time preserve the privacy of owner by sharing information at a certain level of granularity using user policy and dynamic constraints.

VI. Related Work

To enhance sharing of information among collaborating users in dynamic virtual team environment, we extend RBAC model [14] using context constraints, collaborative relationships, owner-defined roles. In addition we use fine-grained context information management [18] and entity priorities. We use all this to cope with the changing requirements of the system and to provide user with information as much as possible. Ad-hoc collaborations in virtual teams are described in [19] and the sharing challenges are described in [15]. In past access control list and capability list were used to implement access control [7]. Scalability of access rights management was a major problem until the development of Role-Based Access Control (RBAC) model [14] and later standardized in [12]. Requirements of access control in collaborative environments are described by [28] and survey is presented in [6]. RBAC has been extended using additional roles in [25] which creates environment roles. Dynamic adaptation of policies for access control systems is described in [36] and [37].

The importance of context-aware systems is highlighted in [20] and [21] and preferences for access control in awareness systems is given in [38]. The DySCon system [39] extends RBAC using context of requester, owner, and environment. In this paper, we extend DySCon to provide hybrid roles and policy creating owner-defined roles that can be revoked using predefined context conditions. Owner-defined roles are also described by [40] in which requester selects one role out of many role provided by owner. It leaves the difficulty of role selection to requester. In our system [35], we use two types of roles, enterprise-defined roles *E-Role* and owner-defined role *O-Role* and provide methods to handle hybrid roles. Conflicts in policies can occur in complex environment where many entities are involved. We use entity priority mechanism and exceptional priorities for handling conflicts. Some strategies for rule conflict handling are described in [9] and [8]. The sharing control system [23] describes a conflict handling mechanism for dynamic collaborative environments using collaborative relationships and priorities of

all entities in the environment.

VII. Conclusion and Future Work

Hybrid roles and collaborative relationships are used in this system to provide enhanced sharing control and privacy to owner's personal and activity related context information. The paper describes user-defined role-based sharing control model, its architecture, and implementation using Web services technology. The RBAC model is extended with context constraints, owner-defined roles *O-Role*, collaborative relationships, and collaboration history. The *O-Roles* assignment and revocation rules are provided. Architecture of the system using Web services and Peer to Peer model is presented and interoperation and working of all components of the architecture is described to provide required level of enhanced sharing and privacy. Due to the involvement of multiple entities and their collaborative relationships, system and its policies becomes complex. Handling of these complexities using extended enterprise-defined RBAC-based policy and user policy based on *O-Role* is described. The evaluation of sharing control request in presence of these hybrid roles and policies is explained using an algorithm. Future work includes the use of autonomic and semantic techniques for dynamic policy adaptation and for description of entities, relationships, and policy rules respectively.

Acknowledgments

This work is partially supported by the Higher Education Commission (HEC) Pakistan and the European Union through the FP7-216256 project COIN.

References

[1] M. Jaume. A formal approach to implement access control models, *Journal of Information Assurance and Security*, 1 (2), pp. 137-148, 2006.

[2] W. Zhou. Authorization Constraints Specification and Enforcement, *Journal of Information Assurance and Security*, 3 (1), pp. 38-50, 2008.

[3] M. Menzel. Access Control for Cross-Organisational Web Service Composition, *Journal of Information Assurance and Security*, 2 (2), pp. 155-160, 2007.

[4] L. Habib. Formal definition and comparison of access control models, *Journal of Information Assurance and Security*, 4 (4), pp. 372-381, 2009.

[5] E. Lupu. Conflicts in policy-based distributed systems management, *IEEE Transactions on Software Engineering*, 25 (6), pp. 852-869, 1999.

[6] W. Tolone, G. Ahn, T. pai. Access control in collaborative systems, *ACM Comput. Surv.*, 37 (1), pp. 29-41, 2005.

[7] R. Sandhu and P. Samarati. Access control: Principles and practice, *IEEE Communications*, 32 (9), pp. 40-48, 1994.

[8] F. Cuppens. High Level Conflict Management Strategies in Advanced Access Control Models, *Electron. Notes Theor. Comput. Sci.*, 186, pp. 3-26, 2007.

[9] S. Jajodia. A unified framework for enforcing multiple access control policies, *SIGMOD Rec.*, 26 (2), pp. 474-485, 1997.

[10] B. Carminati, E. Ferrari. Access control and privacy in web-based social networks, *IJWIS*, 4 (4), pp. 395-415, 2008.

[11] Q. Ni. Privacy-aware role-based access control, *ACM Trans. Inf. Syst. Secur.*, 13 (3), pp. 1-31, 2010.

[12] D. Ferraiolo. Proposed NIST Standard for Role-Based Access Control, *ACM Trans. on Information and System Security (TISSEC)*, 4 (3), pp. 224-274, 2001.

[13] K. Boulos. The emerging Web 2.0 social software: an enabling suite of sociable technologies in health and health care education, *Health Information & Libraries Journal*, 24 (1), pp. 2-23, 2007.

[14] R. Sandhu, E. Coyne, H. Feinstein. C. Youman. Role-Based Access Control Models, *IEEE Computer*, 29 (2), pp. 38-47, 1996.

[15] K. Smith, L. Seligman, V. Swarup. Everybody Share: The Challenge of Data-Sharing Systems, *IEEE Computer*, 41 (9), pp. 54-61, 2008.

[16] A. McAfee. Enterprise 2.0: The Dawn of Emergent Collaboration, *MIT Sloan Management Review*, 47 (3), pp. 21-28, 2006.

[17] N. Eagle, A. Pentland. Social Serendipity: Mobilizing Social Software, *IEEE Pervasive Computing*, 4 (2), pp. 28-34, 2005.

[18] C. Dorn. Sharing hierarchical context for mobile web services, *Distrib. Parallel Databases*, 21 (1), pp. 85-111, 2007.

[19] S. Dustdar. Caramba – A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams, *Journal of Distributed and Parallel Databases*, 15 (1), pp. 45-66, 2004.

[20] M. Baldauf. A survey on context-aware systems, *I-JAHUC*, 2 (4), pp. 263-277, 2007.

[21] H. Truong. A survey on context-aware web service systems, *IJWIS*, 5 (1), pp. 5-31, 2009.

[22] S. Murugesan. Understanding Web 2.0, *IT Professional*, 9 (4), pp. 34-41, 2007.

[23] A. Malik, S. Dustdar. Enhanced sharing and privacy in distributed information sharing environments. In *Proceedings of the 7th International Conference on Information Assurance and Security (IAS)*, pp. 286-291, 2011.

[24] N. Dunlop. Dynamic conflict detection in policy-based management systems. In *Proceedings of the Sixth International Conference on Enterprise Distributed Object Computing (EDOC)*, pp. 15-26, 2002.

[25] M. Convington. Securing context-aware applications using environment roles. In *Proceedings of the 6th ACM SACMAT*, pp. 10-20, 2001.

[26] R. Hulsebosch. Context sensitive access control. In *Proceedings of the 10th ACM SACMAT*, pp. 111-119, 2005.

[27] G. Ahn. Authorization management for role based collaboration. In *Proceedings of the IEEE int. conf. on System, Man and Cybernetic*, pp. 4128-4134, 2003.

[28] H. Shen, P. Dewan. Access control for collaborative environments. In *Proceedings of the ACM conference on Computer-supported cooperative work (CSCW)*, pp. 51-58, 1992.

[29] G. Karjoth, M. Schunter. A Privacy Policy Model for Enterprises. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop (CSFW)*, pp. 271-281, 2002.

[30] B. Carminati, E. Ferrari. Enforcing relationships privacy through collaborative access control in web-based Social Networks. In *Proceedings of the CollaborateCom*, pp. 1-9, 2009.

[31] R. Thomas, R. Sandhu. Task-based authorization controls(TBAC): A family of models for active and enterprize-oriented management. In *Proceedings of the Database Security XI*, pp. 166-181 , 1997.

[32] C. Georgiadis. Flexible Team-based access control using context. In *Proceedings of the ACM SACMAT*, pp. 21-30 , 2001.

[33] R. Thomas. Georgiadis. Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments. In *Proceedings of the second ACM workshop on Role-based access control*, pp. 13-19 , 1997.

[34] A. Kalam. Organization based access control. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, pp. 120-131 , 2003.

[35] A. Malik. Context-aware Sharing Control using Hybrid Roles in Inter-enterprise Collaboration. In *Proceedings of the Fifth International Conference on Software and Data Technologies*, pp. 42-48 , 2010.

[36] Y. Kim. Context-Aware Access Control Mechanism for Ubiquitous Applications. In *Proceedings of the Third International Atlantic Web Intelligence Conference, (AWIC)*, pp. 236-242 , 2005.

[37] A. Toninelli. A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In *Proceedings of the 5th International Semantic Web Conference, (ISWC)*, pp. 473-486 , 2006.

[38] P. Sameer. Who gets to know what when: configuring privacy permissions in an awareness application. In *Proceedings of ACM CHI 2005 Conference on Human Factors in Computing Systems*, pp. 101-110 , 2005.

[39] A. Malik. DySCon: Dynamic Sharing Control for Distributed Team Collaboration in Networked Enterprises. In *Proceedings of the 11th IEEE International Conference on Commerce and Enterprise Computing, (CEC)*, pp. 279-284 , 2009.

[40] C. Groba. Context-Dependent Access Control for Contextual Information. In *Proceedings of ARES Conference*, pp. 155-161 , 2007.

Author Biographies

Dr. Ahmad Kamran Malik is an Assistant Professor at the Institute of Information Technology, Quaid-i-Azam University, Islamabad, Pakistan. He received his PhD in Computer Science from the Vienna University of Technology in 2011. He studied MS in Computer Science at Muhammad Ali Jinnah University, Islamabad, Pakistan and M.Sc. in Computer Science at Gomal University, D.I. Khan, Pakistan. From 1999 to 2007 he has been teaching and supervising computer science students at bachelors and masters level. Currently his research interest is focused on Collaborative systems, Information management and sharing, Information privacy, and Access Control. He has been working on Distributed Databases, Data Integration, Peer data management, and Query processing.

Prof. Schahram Dustdar is Full Professor of Computer Science with a focus on Internet Technologies heading the Distributed Systems Group at Vienna University of Technology, Vienna, Austria. Since 2009 he is an ACM Distinguished Scientist. He received his M.Sc. (1990) and PhD. degrees (1992) in Business Informatics from the University of Linz, Austria. In April 2003 he received his Habilitation degree for his work on Process-aware Collaboration Systems - Architectures and Coordination Models for Virtual Teams. He has published more than 300 scientific papers as conference-, journal-, and book contributions. His research focus is on Internet Technologies and he is head of the Distributed Systems Group. In particular, his interests are in Service-oriented Architectures and Computing, Mobile and Ubiquitous Computing, Complex-, Autonomic-, and Adaptive Systems, and Context-aware Computing including all aspects related to collaborative systems (e.g., Workflow technologies).