Delegating revocations and authorizations in collaborative business environments

Hua Wang [1] Jinli Cao [2] Yanchun Zhang [3]

[1] Department of Maths & Computing

University of Southern Queensland

Toowoomba QLD 4350 Australia

Email: wang@usq.edu.au

[2] Department of Computer Science & Computer Engineering

La Trobe University, Melbourne, VIC 3086, Australia

Email: jinli@cs.latrobe.edu.au

[3] School of Computer Science and Mathematics

Victoria University, Melbourne City, MC8001, Australia.

Email: yzhang@csm.vu.edu.au

Abstract

Efficient collaboration allows organizations and individuals to improve the efficiency and quality of their business activities. Delegations, as a significant approach, may occur as workflow collaborations,

supply chain collaborations, or collaborative commerce. Role-based delegation models have been used as flexible and efficient access management for collaborative business environments. Delegation revocations can provide significant functionalities for the models in business environments when the delegated roles or permissions are required to get back. However, problems may arise in the revocation process when one user delegates user U a role and another user delegates U a negative authorization of the role.

This paper aims to analyse various role-based delegation revocation features through examples. Revocations are categorized in four dimensions: Dependency, Resilience, Propagation and Dominance. According to these dimensions, sixteen types of revocations exist for specific requests in collaborative business environments: DependentWeakLocalDelete, DependentWeakLocalNegative, DependentWeakGlobalDelete, DependentWeakGlobalNegative, IndependentWeakLocalDelete, IndependentWeakLocalNegative, IndependentWeakGlobalDelete, IndependentWeakGlobalNegative, and so on. We present revocation delegating models, and then discuss user delegation authorization and the impact of revocation operations. Finally, comparisons with other related work are discussed.

1 Introduction

Business process modelling describes how activities interact and relate with other organizations while supporting business operations in a collaborative environment. Collaboration links organizations and individuals together to improve the efficiency of commercial business such as sales, procurement,

manufacturing, distribution and replenishment. Collaboration has moved beyond mere buying and selling to encompass planning, design, development, communication, the sourcing of information, research, and the provision of services among organizations [7, 14]. Collaborative business is an application of information technology to achieve a closer integration and a better management of business relationships among internal and external parties. Significant work has been done on developing business process management, methodologies and ontologies as well as on the specification of process modelling languages [5, 30]. There are situations in which collaboration resources cannot be updated or delivered due to insufficient collaborative management arrangements between business partners. It is still an open question how efficiently technical skills can be trained in collaborative business environments [4, 19].

Therefore, effective and efficient communication with distant collaborators is required for business collaborations. This paper aims to develop role-based delegation models for collaborative business. The inclusion of role-based delegation and revocation allows users themselves to delegate role authorities to others to process some authorized functions and later remove those authorities. Role-based delegation and revocation models are developed with comparison to established technical analysis, laboratory experiments, support hierarchical roles and multistep delegation. The models are implemented to demonstrate their feasibility and secure protocols for managing delegations and revocations.

Delegation is the process whereby an active entity grants access resource permissions to another entity. The basic idea of delegation is to enable someone to do a job, for example, a secretary. Effective delegation not only makes management systems ultimately more satisfactory, but also frees the delegating users to focus on other important issues. In access control management systems, the delegation arises when users need to act on another user's behalf in accessing resources. Delegation is recognised as vital in a secure distributed computing environment [1, 2]. Delegation is an important feature in many business collaborations. For example, the Immigration Department

is developing partnerships between immigration agencies and people in local areas to address possible problems. Immigration officers are able to prevent illegal stay and crime if they efficiently collaborate with these people. The problem-oriented immigrating system (*POIS*) is proposed to improve the service as a part of the Immigration Department's ongoing community efforts, including identifying potential problems and resolving them before they become significant. With efficient delegation, officers respond quickly to urgent messages and increase the time spent confronting problems. On the other hand, it is natural for a user to revoke a delegated role that he/she granted to other users. A revoke operation may in turn reduce the management processes and makes the delegating system more efficient. Many revocation models have been proposed in the past a few years [6, 9, 27], and the result of revocation operations depends on the particular definitions of the operation in the models.

The *NIST* developed role-based access control (*RBAC*) prototype [10] and published a formal model [11]. The basic elements and relations in role-based access control (*RBAC*) are depicted in Figure 1. *RBAC* enables managing and enforcing security in large-scale and enterprise-wide systems. In *RBAC* models, permissions are associated with roles, users are assigned to appropriate roles, and users acquire permissions through roles. Users can be easily reassigned from one role to another. Roles can be granted new permissions and permissions can be revoked from roles as needed. Therefore, *RBAC* provides a means for empowering individual users through role-based delegation in distributed collaboration environments. However, there is little work on delegation revocations with *RBAC*. We analyse various delegation revocations based on the *RBAC* model.

Revocation is a significant function in role-based delegations. For example, *Tony* delegated role director (*DIR*) to *Richard*; if *Richard* moves to another company and does not work at the Immigration Department, his delegated role *DIR* has to be revoked instantly. Several different semantics of user revocation exist [13]: global and local (propagation), strong and weak (dominance) and deletion or negative (resilience). Propagation refers to the

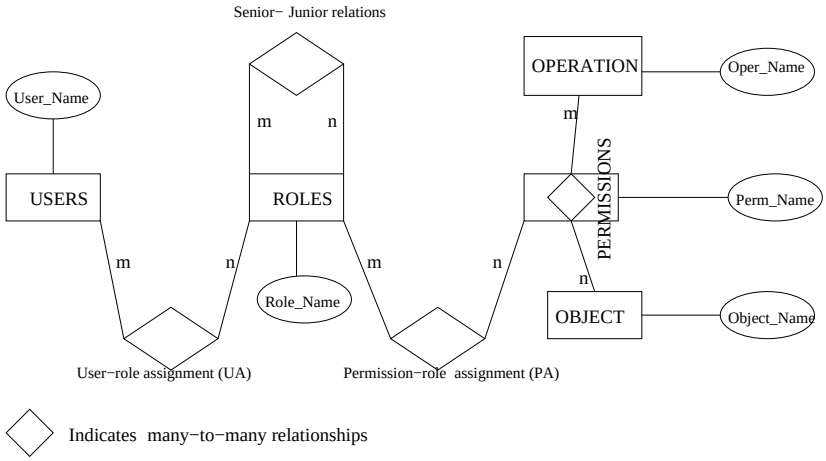

Figure 1: RBAC relationship

extent of the revocation to other delegated users while Resilience means no time-persistent with negative permissions. Dominance refers to the effect of a revocation on implicit/explicit role memberships of a user. For example, there are two types of revocation in dominance: weak and strong revocation [21]. A strong revocation of a user from a role requires that the user be removed not only from the explicit membership but also from the implicit memberships of the delegated role. A weak revocation only removes the user from the delegated role (explicit membership) and leaves other roles intact. Strong revocation is theoretically equivalent to a series of weak revocations. To perform strong revocation, the implied weak revocations are authorized based on revocation policies.

The purpose of the paper is to describe and analyse a number of revocation schemes and their relationships to one another. Revocation schemes are categorized with four dimensions: dependency, resilience, propagation and dominance. With the help of these dimensions we define sixteen different revocation schemes. The remainder of this paper is organized as follows: Section 2 presents the required technologies for the paper. It includes a delegation example, *RBAC* and role-based delegation structure. Section 3

proposes the details of four revocation dimensions. Rather than formal definition, examples are used to explain the definitions of each dimension. There are sixteen types of revocation based on the dimensions. We do not analyse all sixteen revocation schemes in the paper, but, four of them are discussed in Section 4. Section 5 describes the implementation of the role-based delegating revocations using *XML* technology and Section 6 compares our work with the previous work on delegation revocation methods. The differences between the work in this paper and others are presented. Section 7 concludes the paper and outlines our future work.

2 Basic technologies

2.1 Delegations

In problem-oriented immigrating system (*POIS*), officers might be involved in many concurrent activities such as conducting initial investigations, analysing and confronting crimes, preparing immigration reports, and assessing projects. In order to achieve this, users may have one or more roles such as lead officer, participant officer, or reporter. In this example, *Tony*, a director, needs to coordinate analysing and confronting crimes and assessing projects. Collaboration is necessary for information sharing with members from these two projects. To collaborate closely and make two projects more successful, *Tony* would like to delegate certain responsibilities to *Christine* and her staff. The prerequisite conditions are to secure these processes and to monitor the progress of the delegation. Furthermore, *Christine* may need to delegate the delegated role to her staff as necessary or to delegate a role to all members of another role at the same time. Without delegation capacity, security officers have to do excessive work since they are involved in every single collaborative activity. We can find the major requirements of role-based delegation in this example:

1. Group-based delegation means that a delegating user may need to delegate a role to all members of another role at the same time.

2. Multistep delegation occurs when a delegation can be further delegated. Single-step delegation means that the delegated role cannot be further delegated.

3. Revocation schemes are important characteristics of collaboration. They take away the delegated permissions. There are different revoking schemes, among them are strong and weak revocations, cascading and noncascading revocations, as well as grant-dependent and grant-independent revocations [21].

4. Constraints are an important factor in *RBAC* for laying out higher-level organizational policies [20]. They define whether or not the delegation or revocation process is valid.

5. Partial delegation means only subsets of the permissions are delegated while total delegation means all permissions are delegated. Partial delegation is an important feature because it allows users to only delegate required permissions. The well-known Least Privilege Security Principle can be implemented through partial delegation.

This paper focuses exclusively on revocation schemes in role-based delegation models. We extend our previous work and propose a delegation framework and analyse how original role assignment changes impact delegation results.

2.2 Basic elements and components

RBAC involves individual users being associated with roles as well as roles being associated with permissions (Each permission is a pair of objects and operations). As such, a role is used to associate users and permissions. A user in this model is a human being. A role is a job function or job title within

the organization associated with authority and responsibility [8, 24]. RBAC is being considered as part of the emerging SQL3 standard for database management systems, based on their implementation in Oracle 7 [16, 18]. Many RBAC practical applications have been implemented [3, 17].

A session is a mapping between a user and possibly many roles. For example, a user may establish a session by activating some subset of assigned roles. A session is always associated with a single user and each user may establish zero or more sessions. There may be hierarchies within roles. Senior roles are shown at the top of the hierarchies. Senior roles inherit permissions from junior roles. Let $x > y$ denote x is senior to y with obvious extension to $x \geq y$. Role hierarchies provide a powerful and convenient means to enforce the principle of Least Privilege since only required permissions to perform a task are assigned to the role.

Figure 2 shows the role hierarchy structure of *RBAC* in *POIS*. A high location role is senior to a low connected location role in the Figure, e.g. role *Co1* is senior to role *AP*.

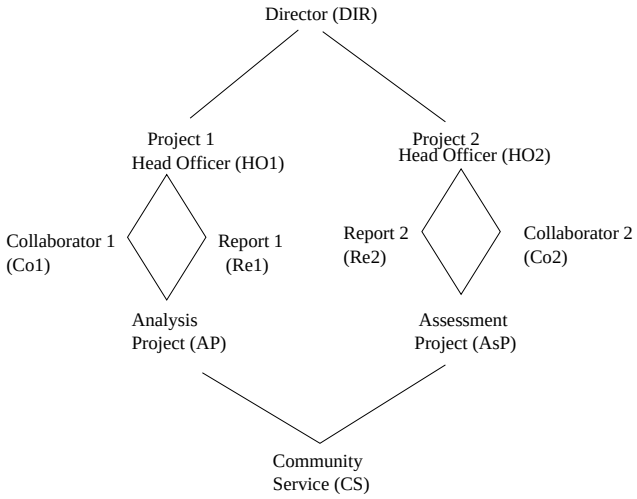

Figure 2: Role hierarchy in *POIS*

The following Table 1 expresses an example of user-role assignment in *POIS*.

RoleName	UserName
DIR	Tony
HO1	Richard
HO2	Mike
Co1	Sam
Re1	Christine
CS	Ahn

Table 1: User-Role relationship

There are two sets of users associated with role r:

Original users are those users who are assigned to the role r;

Delegated users are those users who are delegated to the role r.

The same user can be an original user of one role and a delegated user of another role. Also it is possible for a user to be both an original user and a delegated user of the same role. For example, if *Richard* delegates his role *HO1* to *Sam*, then *Sam* is both an original user (explicitly) and a delegated user (implicitly) of role *Co1* because the role *HO1* is senior to the role *Co1*. The original user assignment (UAO) is a many-to-many user assignment relation between original users and roles. The delegated user assignment (UAD) is a many-to-many user assignment relation between delegated users and roles.

We have the following components for role-based delegation model ($RBDM$):

U, R, P and S are sets of users, roles, permissions, and sessions, respectively.

1. $UAO \subseteq U \times R$ is a many-to-many original user to role assignment relation.

2. $UAD \subseteq U \times R$ is a many-to-many delegated user to role assignment relation.

3. $UA = UAO \cup UAD$.

4. Users: $R \Rightarrow 2^U$is a function mapping each role to a set of users. $Users(r) = \{u|(u,r) \in UA\}$ where UA is user-role assignment.

5. $Users(r) = Users\_O(r) \cup Users\_D(r)$
   where
   $Users\_O(r) = \{u|\exists r' > r, (u,r') \in UAO\}$,
   $Users\_D(r) = \{u|\exists r' > r, (u,r') \in UAD\}$.

$Users(r)$ includes all users who are members of role r. The users may be original and delegated users. The original users $Users\_O(r)$ are not only the member of role r but also the member of a senior role r' of r. This is because the senior role r' inherits all permissions of its junior role r. With the hierarchical structure of roles, a user who is a member of r' is also a member of role r. The members in $Users\_D(r)$ are similar to that in $Users\_O(r)$. For instance, according Table 1 and the example of *Richard* delegating role *HO1* to *Christine*, we have:
$Users\_O(HO1) = \{Richard\}$,
and
$Users\_D(HO1) = \{Christine\}$.

2.3 Role-based delegation and revocation

The scope of our model is to address user-to-user delegation supporting role hierarchy. We consider only the regular role delegation in this paper, even though it is possible and desirable to delegate an administrative role.

A delegation relation ($DELR$) exists in the role-based delegation model which includes three elements: original user assignments UAO, delegated

user assignment UAD and constraints. The motivation behind this relation is to address the relationships among different components involved in a delegation. In a user-to-user delegation, there are five components: a delegating user, a delegating role, a delegated user, a delegated role, and associated constraints. $((Tony, DIR), (Mike, DIR), Friday)$, for example, means *Tony* acting in role *DIR* delegates role *DIR* to *Mike* on Friday. We assume each delegation is associated with zero or more constraints. The delegation relation supports partial delegation in a role hierarchies: a user who is authorized to delegate a role r can also delegate a role r' that is junior to r. For example, $((Tony, DIR), (Ahn, Re1)$, Friday) means *Tony* acting in role *DIR* delegates a junior role *Re1* to Ahn on Friday. A delegation relation is one-to-many relationship on user assignments. It consists of original user delegation ($ORID$) and delegated user delegation ($DELD$). Figure 3 illustrates components and their relations in a role-based delegation model.

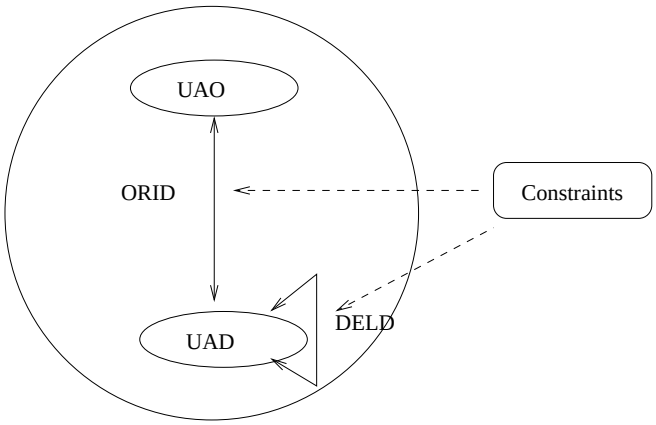

Figure 3: Role-based delegation model

From the above discussions, the following components are formalized:

1. $DELR \subseteq UA \times UA \times Cons$ is one-to-many delegation relation. A delegation relation can be represented by $((u, r), (u', r'), Cons) \in DELR$, which means the delegating user u with role r delegated role r' to user u' who satisfies the constraint $Cons$.

2. $ORID \subseteq UAO \times UAD \times Cons$ is an original user delegation relation.

3. $DELD \subseteq UAD \times UAD \times Cons$ is a delegated user delegation relation.

4. $DELR = ORID \cup DELD$.

We find from the components above that delegation relations include original user delegations and delegated user delegations. There are related subtleties concerning the interaction between delegation and revocation of user-user delegation membership and the role hierarchy.

Definition 1 A user-user delegation revocation is a relation $Can - revoke \subseteq R \times 2^R$, where R is a set of roles. ◇

The meaning of Can-revoke (x, Y) is that a member of role x (or a member of an role that is senior to x) can revoke delegation relationship of a user from any role $y \in Y$, where Y defines the *range of revocation*. Table 2 gives the Can-revoke relation in Figure 2.

RoleName	Role Range
HO1	[Co1, CS]

Table 2: Can-revoke

We analyse the revocation dimension in the next section followed by a rich types of revocation schemes.

3 Revocation dimensions

Different revocation models have been proposed for access control systems [13, 28]. For instance, three dimensions were introduced in [13] that are applied for database management system. We gather these dimensions in a unified collections: dependency, resilience, propagation and dominance. Each dimension is defined in this section adopting examples rather than formal descriptions.

3.1 Dependency

Dependency refers to the legitimacy of a user who can revoke a delegated role. Dependent revocation means only the delegating user can revoke the delegated user from the delegated role. Independent revocation allows any original user of the delegated role can revoke the user from the delegated role. For example, with dependent revocation *Richard* can revoke *Sam* from the delegated role *Co1* but *Tony* cannot, even though *Tony* acts as role *DIR* that is senior to and an original role of *Co1*, but *Tony* can take the delegated role *Co1* from *Richard* with independent revocation.

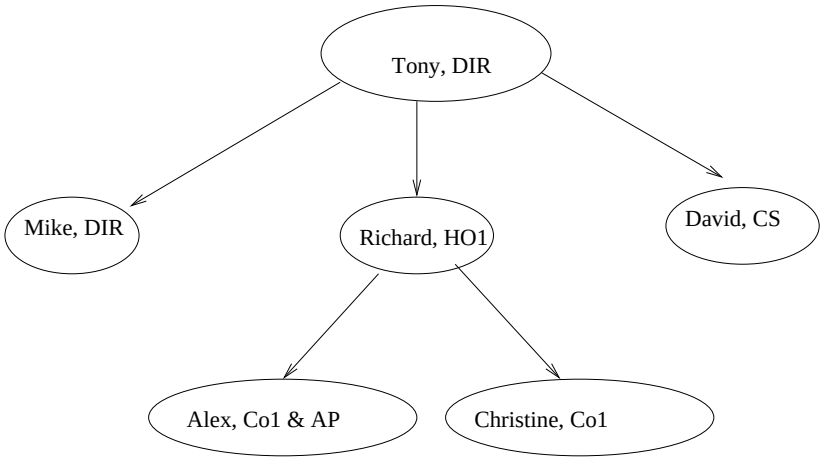

Figure 4: Revocation relationships

3.2 Resilience

This dimension distinguishes revocation through deleting or negative authorization. The effect of a role deleting revocation from a user is acted; another user may grant the user the same role that was just revoked, and as a result the revocation has no affect on the user. The negative authorization has high priority in this dimension that means the authorization overrule any other authorizations until the negative one is in turn revoked. As shown in

Figure 4, *Richard* needs to keep *Alex* from role *Co1*, she can either delete the current delegating relationship, or give a negative authorization of *Co1* to *Alex*. In the first case, *Alex* is denied to act as role *Co1*, but only as long as no other users delegate *Alex* the role. In the second case, *Alex* can not act as role *Co1* until the negative authorization is revoked. Therefore, negative authorization is stronger than deleting revocation.

3.3 Propagation

This dimension distinguishes revocations according to a delegation structure of role locations. There are local revocations and global revocations in this dimension. A local revocation only applies to the direct delegation relationship while the global revocation effects all other users authorized by the revoked user. We use the Figure 4 to explain the difference between local and global revocations. Suppose *Tony* wants to revoke *Richard* from *HO1* but trusts *Alex* with roles *AP, Co1*, and trust *Christine* with role *Co1* delegated by *Richard*, local revocation can be applied; otherwise global revocation is applied to take role *HO1* from *Richard* as well as roles *AP, Co1* from both *Alex* and *Co1* from *Christine*.

3.4 Dominance

This dimension deals with role structure in *RBAC*. Due to role hierarchy, a role x' has all permissions of role x when x' is senior to x ($x' > x$). A revocation problem may arise when a user with two roles $\{x', x\}$, the user still has the permissions of x if only to revoke x from the user. The explicit member of a role x is a set of user $\{U|(U, x) \in UA\}$ where $(U, x) \in UA$ means user U has role x and the implicit member of role x is a set of user $\{U|\exists x' > x, (U, x') \in UA\}$. To solve the authorization revocation problem, we need to revoke the explicit member of a role first if a user is an explicit member, then revoke the implicit member. There are two kinds of revocations. The first

one is weak revocation which revokes explicit member only; the second one is strong revocation that revokes explicit and implicit members. *Alex* has two delegated roles *Co1, AP* where role *Co1* is senior to role *AP* in Figure 4. *Richard* wants to take role *AP* from *Alex* with strong revocation, both roles *Co1* and *AP* are revoked from *Alex*, but with weak revocation only role *AP* is revoked.

4 Delegating revocations

We describe sixteen different revocations based on the dimensions in the previous section. Each scheme, as shown in Table 3, has a unique description with respect to the four dimensions. Each revocation scheme in the table is important because everyone exists in our real life. We only analyze four delegating revocations in this paper due to the length of the paper and provide what environments they may apply.

4.1 DependentWeakLocalDelete(*DWLD*)

The DependentWeakLocalDelete (*DWLD*), as the first revocation scheme, is the most easy operation. It has neither resilience, propagation, nor dominance, and only the direct delegating user can remove the delegation relationship. Suppose *Tony* as the director wants to revoke role *HO1* from *Richard* since *Richard* is not an employee any longer in a company. With the scheme of *DWLD*, role *Ho1* only takes away from *Richard* and roles of both *Alex* and *Christine* are intact; the delegation relationships between *Richard* and *Alex*, *Richard* and *Christine* come to the relationships between *Tony* and *Alex*, *Tony* and *Christine* as shown in Figure 5 from Figure 4. The features of scheme *DWLD* when user U_1 wants to revoke role r from user U_2 are:

1. U_1 does not grant role r to U_2;

No.	Dependency	Resilience	Propagation	Dominance	Name
1	No	No	No	No	DependentWeakLocalDelete(*DWLD*)
2	No	No	No	Yes	DependentStrongLocalDelete(*DSLD*)
3	No	No	Yes	No	DependentWeakGlobalDelete(*DWGD*)
4	No	No	Yes	Yes	DependentStrongGlobalDelete(*DSGD*)
5	No	Yes	No	No	DependentWeakLocalNegative(*DWLN*)
6	No	Yes	No	Yes	DependentStrongLocalNegative(*DSLN*)
7	No	Yes	Yes	No	DependentWeakGlobalNegative(*DWLN*)
8	No	Yes	Yes	Yes	DependentStrongGlobalNegative(*DSGN*)
9	Yes	No	No	No	IndependentWeakLocalDelete(*IDWLD*)
10	Yes	No	No	Yes	IndependentStrongLocalDelete(*IDSLD*)
11	Yes	No	Yes	No	IndependentWeakGlobalDelete(*IDWGD*)
12	Yes	No	Yes	Yes	IndependentStrongGlobalDelete(*IDSGD*)
13	Yes	Yes	No	No	IndependentWeakLocalNegative(*IDWLN*)
14	Yes	Yes	No	Yes	IndependentStrongLocalNegative(*IDSLN*)
15	Yes	Yes	Yes	No	IndependentWeakGlobalNegative(*IDWLN*)
16	Yes	Yes	Yes	Yes	IndependentStrongGlobalNegative(*IDSGN*)

Table 3: Revocation types

2. Role r may still stay with U_2 if users other than U_1 delegate r to him;
3. Roles granted by users other than U_1 are intact;
4. The delegation structure may need to update since roles delegated by U_2 have to remain.

The revocation scheme *DWLD* is happened very often in real world. It is able to apply in any collaborative workplace due to the relationship is not changed except the person left the workplace.

4.2 DependentStrongLocalDelete(*DSLD*)

The DependentStrongLocalDelete (*DSLD*) is different from *DWLD* in the dominance aspect. It does not have resilience or propagation, but does have

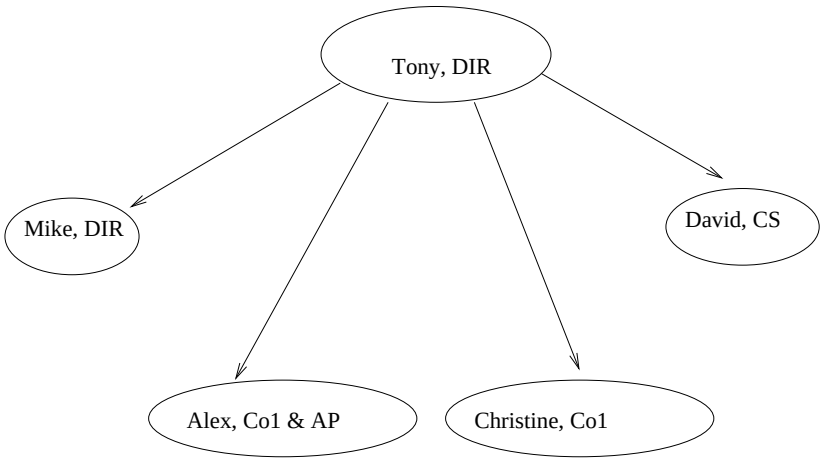

Figure 5: Delegation relationships after *DWLD*

dominance, and only the direct delegating user can take away the delegation relationship. Suppose *Mike* acts as role *DIR* wants to remove role *Co1* from *Richard* since *Richard* is out of the work in role *Co1*. With the scheme of *DSLD*, not only role *Co1* is removed from *Richard* by *Mike*, but role *HO1* is also removed from *Richard* by *Tony* since role *HO1* delegated by *Tony* is senior to role *Co1*. New delegation relationships are generated between *Mike* and *Alex*, *Mike* and *Christine* as shown in Figure 6.

The features of scheme *DSLD* when user U_1 wants to revoke role r from user U_2 are:

1. U_1 does not grant role r to U_2;
2. Implicit role r' that is senior to role r is removed;
3. Roles other than r and its senior role are intact;
4. The delegation structure may need to be updated since roles delegated by U_2 have to remain.

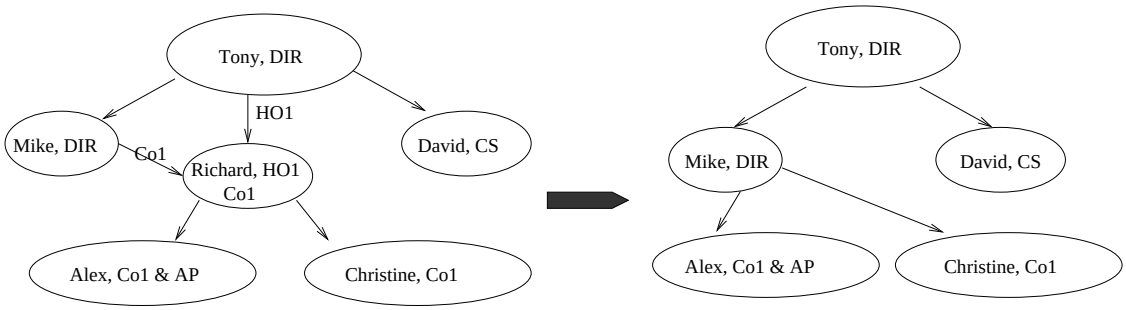

Figure 6: Delegation relationships after *DSLD*

The revocation scheme *DSLD* is happened when a user in the collaborative environment has more than two roles, and one of the two roles comes from another organisation that has to be removed by the direct delegating user. It is a good solution to reject a user (e.g. *Richard*) acting a special role (e.g. *Co1*).

4.3 DependentWeakGlobalDelete(*DWGD*)

The DependentWeakGlobalDelete (*DWGD*) is different from *DWLD* in the propagation aspect. It does not have resilience and dominance, but propagation, and only the direct delegating user can remove the delegation relationship. Suppose *Mike* acts as role *DIR* wants to remove role *Co1* from *Richard* since *Richard* is out of the work in role *Co1*. With the scheme of *DWGD*, role *Co1* is removed from *Richard* by *Mike*, but role *HO1* is intact, role *Co1* is also revoked from *Alex* and *Christine* since the delegation authorization is no longer supported by *Richard*. The new relationships after the *DWGD* are shown in Figure 7.

The features of scheme *DWGD* when user U_1 wants to revoke role r from user U_2 are:

1. U_1 does not grant role r to U_2;

2. role r delegated by U_2 to users is removed;

3. Roles other than r are intact;

4. The delegation structure may need to be updated since roles other than r delegated by U_2 have to remain.

The revocation scheme *DWGD* is happened more strict than the revocation scheme *DSLD*. It can successfully protect subordinates of a user (e.g. *Richard*) obtaining an access permission (e.g. *Co1*). It may be used in military or security environments.

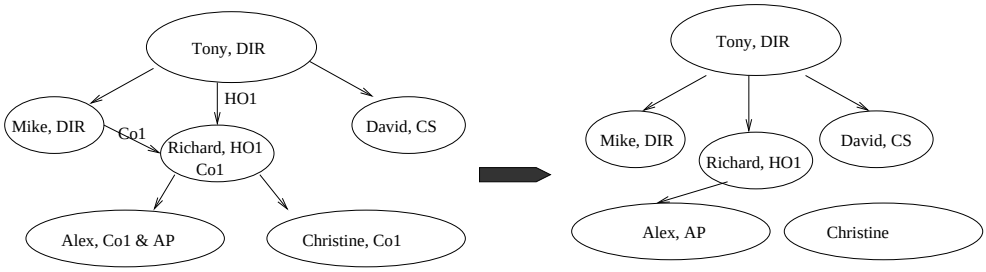

Figure 7: Delegation relationships after *DWGD*

4.4 DependentWeakLocalNegative(*DWLN*)

The DependentWeakLocalNegative (*DWLN*) is the scheme with negative authorization in the resilience dimension. When a negative authorization of a role is granted to a revokee as a weak and local revocation, the negative authorization will remain with the revokee until the authorization is removed, and the revokee cannot act in the role even though the role is delegated to her/him. The negative revocation scheme is different from other revocation schemes in the resilience dimension. We use the black color for the role with negative authorization in Figure 8. Suppose *Mike* acts as role *DIR* wants to grant a negative authorization of role *Co1* to *Richard* since *Richard* is no

longer loyal in role *Co1*. With the scheme of *DWLN*, role *Co1* takes away from *Richard* by *Mike*, but role *HO1* is intact. Role *Co1* is not revoked from *Alex* and *Christine* since the revocation is weak.

The features of scheme *DWLN* when user U_1 wants to revoke role r from user U_2 as a negative authorization are:

1. role r delegated to U_2 is inactive;
2. roles other than r delegated by U_2 are intact;
3. Roles other than r delegated to U_2 are intact;
4. The delegation structure may need to be updated since roles other than r delegated by U_2 have to remain.

The revocation scheme *DWLN* is used in a different way from the revocation scheme *DWGD* where a user (e.g. *Richard*) is no longer loyal in a role (e.g. *Co1*) and the user cannot act as the role, however his/her subordinates still have the role. Therefore, the subordinates are trusted by the system. This revocation scheme may be happened in general collaborative systems such as education environment.

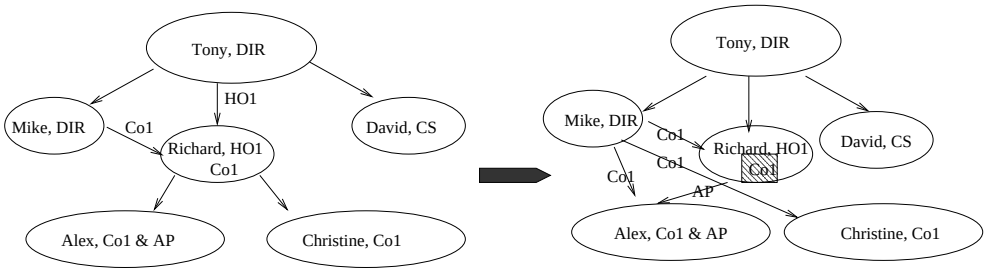

Figure 8: Delegation relationships after *DWLN*

Each type of revocation scheme in Table 3 is as same important as others. We have analysed four revocations in this section but not for the other revocation operations due to the length of the paper.

5 XML implementation

This section presents the implementation of the group delegation with *XML*-based technology. XML has many advantages: simplicity, entensiblity, interoperability and so on. The reason of using XML is that we do not need to worry about the data structure of each organisation in collaborative environments. It is easy to add nodes to the existing XML documents when another organisation is required to join the business environment in future. This is a web-based project implemented in XAMPP (http://www.apachefriends.org/en/xampp.html) environment in windows platform using XML and XUL (http://www.xulplanet.com/). XML is used to store the information in the collaborative business environment. XUL is Mozilla's XML-based user interface language and is designed to be platform-neutral, making applications easily portable to all of the operating systems on which Mozilla runs. XML DOM is used to deal with every node of XML file. All these are server-side scripting pages. Hence all these pages are stored in the root directory of web-server.

The format of a role-based delegation from the previous section is $((u, r), (u', r'), Cons)$. To maintain the relationship between roles, we define senior and junior roles. For convenience we use the file *rolehie.xml* to store the role hierarchy of the children and parent roles. Based on Table 1, a part of role hierarchy of Figure 2 is modelled in *rolehie.xml* using *IDREF* attributes [15] as shown in Table 4. The hierarchy is not a tree but a graph, for clarity and conciseness. Table 4 shows the hierarchy as a nested relation. The first column gives a role name, the second column gives the immediate senior roles of that role, and the third column gives the immediate junior roles. The ϕ means that the role has no parent or child as the case may be. Using *rolehie.xml*, we can find all seniors and juniors for a group by respectively identifying the parents and children using simple *XPath* query expressions. An example of the role hierarchy of Figure 2 is represented in *rolehie.xml* using *IDREF* attributes as shown in Table 4.

Role Name	Senior role	Junior role
DIR	ϕ	HO1, HO2
HO1	DIR	Co1, Re1
HO2	DIR	Co2, Re2
Co1	HO1	AP
Re1	HO1	AP
AP	Co1, Re1	CS
CS	AP, AsP	ϕ

Table 4: Role hierarchy of Figure 2.

Many XML schemas exist in the implementation such as user, role, user-role assignment, revocation types schema. For example, revocation types schema in Table 5. The revocation types schema in Table 5 explains structure of revocation types in Table 3. All the revocation types we have are included in enumeration block. This shows these are the only available revocation types. Attributes revocationID is the ID attribute and is required. Remaining attributes dependency, resilience, propagation and dominance are Boolean types and are also required.

There is a procedure for revocating a role from a user in our implementation. The procedure call is *Revoke(role, user)*. The parameters role and user specify which role is to be revoked from a user. The revocation function has the following main steps: 1) Select a role to be revoked (or delegated); 2) Select a user to revoke (or delegate); 3*) Check whether or not the role satisfies the delegating authorization rules (only for delegation); 4*) Setup constraints (only for delegation); 5) Select a revocation model; and 6) Update the *DELR* database. For delegation revocation, instead of the steps 3 and 4, one must check whether or not the revocation authorization rules are successful and the revoked role is in the role range of the relation Can_revoke.

<xs:element name="RevocationTypes" >

<xs:complexType >

<xs:restriction base="xs:string" >

<xs:enumeration value="Dependent Weak Local Delete" / >

<xs:enumeration value="Dependent Strong Local Delete" / >

...

<xs:enumeration value="Independent Strong Global Negative" />

</xs:restriction >

<xs:attribute name="revocationID" type="xs:ID" use="required" / >

<xs:attribute name="dependency" type="xs:boolean" use="required" / >

<xs:attribute name="resilience" type="xs:boolean" use="required" / >

<xs:attribute name="propagation" type="xs:boolean" use="required" / >

<xs:attribute name="dominance" type="xs:boolean" use="required" />

Table 5: *XML* Schema for Revocation Types

The *DELR* database maintains role hierarchy information (*rolehie.xml*), explicit membership (*explicit.xml*), and the $Can\_delegate$ (for delegation) and Can_revoke relation tables.

In order to make our implementation more convenient we developed a graphical user interface which interacts with this procedure to do role-based delegation and revocations. The graphical user interface is illustrated in Figure 9. This interface was developed using *XUL* and is used to initiate role-based delegation instead of typing the above procedure call. This implementation is convenient for users since they only need to define the role hierarchy and the relation Can_revoke.

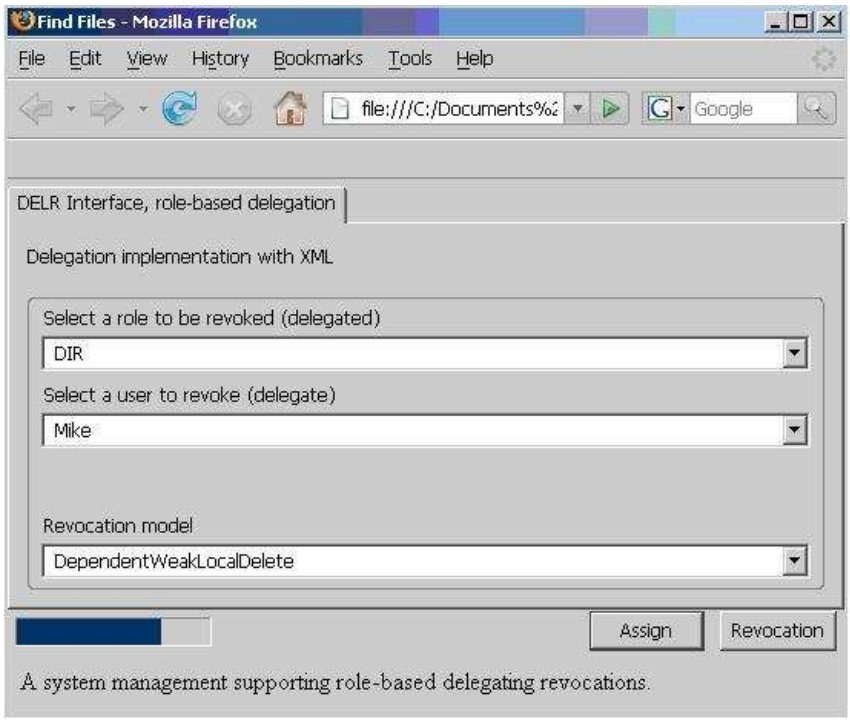

Figure 9: Role-based delegating revocation interface.

6 Comparisons

There are related works on revocation schemes. The closest works to this paper are a rule-based framework for role-based delegation and revocation [29] and revocations-a classification [13]. The former one was on the delegation framework and the latter one worked on database systems.

Zhang et. al [29] presented a role-based delegation model called RDM2000 supporting hierarchical roles and multistep delegation. Different approaches for delegation and revocation were explored. A rule-based language for specifying and enforcing policies on RDM2000 was proposed. The authors described a proof-of-concept prototype implementation of RDM2000 to demonstrate the feasibility of the proposed framework and provide secure protocols for managing delegations. Revocation schemes analysed in this paper in-

cluded three dimensions: Grant-dependency, Propagation and Dominance. There are eight types of revocations according these dimensions. The authors mainly shown grant-dependent revocation. By contrast, the work in this paper provides a rich variety of options that can deal with delegation revocations and the effects after each revocation operation is processed. The work in this paper not only includes the comprehensive revocation schemes but provides more significant work on delegation revocation models.

Hagstrom et. al [13] have argued that several different semantics are possible for the revoke operation that focus on database systems. Three main revocation characteristics were identified: the extent of the revocation to other grantees (propagation), the effect on other grants to the same grantee (dominance), and the permanence of the negation of rights (resilience). A classification was devised using these three dimensions. However, dependency was not included in this paper and hence only eight different revocations were discussed. Their work is different from ours in two aspects. First, it focuses on database authorization. No hierarchical structure of database systems was analysed. As a result, important features such as role hierarchies and constraints were not supported. By contrast, our work focuses on role-based access model that supports hierarchical structure. Second, they did not discuss either who has the ability to process a revocation operation, which is a critical notion to the delegation model, or the relationships among original user and delegated user. By contrast, the delegation model in this paper is based on original user and delegated user since the delegating relationship in this paper has five components $((u, r), (u', r'), Cons)$ in which (u, r) is original user-role relationship and (u', r') be delegated user-role relationship.

Firozabadi and Sergot [12] extended a framework to support a richer set of revocation schemes. The framework was designed for updating privileges and creating management structures by means of authority certificates, and used to create access-level permissions and to delegate authority to other agents. The revocation schemes were based on three separate dimensions: resilience, propagation, and dominance. The authors concerned with the revocation of

certificates, and recognised that only the propagation and dominance dimensions were able to apply to the revocation schemes of the framework and the resilience dimension could not apply. The work in this paper analysed the possible revocation schemes for the existing framework from a real application. This is a good example that demonstrates the differences between the theory of delegation revocation and their real applications. By contrast, we have shown the various revocation schems from theoretical aspect. This is the basic difference between the work in [12] and the work in this paper. The revocation schemes in [12] were built on three dimensions but four dimensions are discussed for delegation revocation in our work. Our work in this paper focuses on detailed theory analysis. Delegation revocation schemes used in applications are much more simple than that in theoretical analysis.

7 Conclusions and future work

This paper has discussed role-based delegation, revocation and authorization. We have discussed role-based delegation requirements and components in delegation models, and analysed not only revocation dimensions but also a variety of revocation schemes. To introduce a practical example of how to use these revocation ideas, a briefly introduction to revocation authorization approach was presented. The theory in this paper was demonstrated by its implementation with *XML*. The work in this paper has significantly extended previous work in several aspects. For example, comprehensive revocation dimensions and revocation schemes for various requirements are analysed. The work in this paper is helpful for building a management system with different revocations.

This is the beginning of work on revocation of role-based access control. The future work will include developing algorithms based on the dimensions and revocation schemes and delegation revocation models with constraints.

References

[1] Abadi M., Burrows M., Lampson B., and Plotkin G. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4):706–734, 1993.

[2] Barka E. and Sandhu R. Framework for role-based delegation models and some extensions. In *Proceedings of the 16 Annual Computer Security Applications Conference*, pages 168–177, New Orleans, 2000a.

[3] Barkley J. F., Beznosov K. and Uppal J. Supporting relationships in access control using role based access control. In *Third ACM Workshop on RoleBased Access Control*, pages 55–65, October 1999.

[4] Bertino E., Crampton J. and Paci F. Access control and authorization constraints for ws-bpel. In *ICWS '06: Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, pages 275–284, Washington, DC, USA, 2006. IEEE Computer Society.

[5] Bertino E., Ferrari E. and Atluri V. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, 1999.

[6] Bertino E. and Jajodia S. and Samarati P. A non-timestamped authorization model for data management systems. In *ACM Conference on Computer and Communications Security*, pages 169–178, 1996.

[7] Caetano A., Zacarias M., Silva A. and Tribolet J. A role-based framework for business process modeling. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 1*, page 13.3, Washington, DC, USA, 2005. IEEE Computer Society.

[8] David F. F., Dennis M. G. and Nickilyn L. An examination of federal and commercial access control policy needs. In *NIST NCSC National Computer Security Conference*, pages 107–116. Baltimore, MD, September 1993.

[9] Fagin R. On an authorization mechanism. *ACM Trans. Database Syst.*, 3(3):310–319, 1978.

[10] Feinstein H. L. Final report: Nist small business innovative research (sbir) grant: role based access control: phase 1. technical report. In *SETA Corp.*, 1995.

[11] Ferraiolo D. F. and Kuhn D. R. Role based access control. In *15th National Computer Security Conference*, pages 554–563. ferraiolo92rolebased.html, 1992.

[12] Firozabadi B. and Sergot M. Revocation Schemes for Delegated Authorities. In *Proceeding of Policy 2002*, http://www.citeseer.ist.psu.edu/firozabadi02revocation.html.

[13] Hagstrom, A., Jajodia, S., Presicce, F., and Wijesekera, D. Revocations-a classification. In *Proceedings of 14th IEEE Computer Security Foundations Workshop*, pages 44–58. Nova Scotia, Canada, 2001.

[14] Li E., Du T. and Wong J. Access control in collaborative commerce. *Decis. Support Syst.*, 43(2):675–685, 2007.

[15] Michael H. *XSLT Programmer's Reference*. Wiley, 2001.

[16] Sandhu R. Rational for the *RBAC*96 family of access control models. In *Proceedings of 1st ACM Workshop on Role-based Access Control*, pages 64–72. ACM Press, 1997.

[17] Sandhu R. Role activation hierarchies. In *Third ACM Workshop on RoleBased Access Control*, pages 33–40. ACM Press, October 1998.

[18] Sandhu R. Role-Based Access Control. *Advances in Computers*, 46, 1998.

[19] Stafford, T.F. Understanding motivations for internet use in distance education. *IEEE Transactions on Education*, 48(2):301–306, 2005.

[20] Wang H., Cao J. and Zhang Y. A consumer anonymity scalable payment scheme with role based access control. In *2nd International Conference on Web Information Systems Engineering (WISE01)*, pages 53–62, Kyoto, Japan, December 2001.

[21] Wang H., Cao J. and Zhang Y. Formal authorization allocation approaches for role-based access control based on relational algebra operations. In *3nd International Conference on Web Information Systems Engineering (WISE02)*, pages 301–312, Singapore, December 2002.

[22] Wang H., Cao J. and Zhang Y. Formal authorization allocation approaches for permission-role assignments using relational algebra operations. In *Proceedings of the 14th Australian Database Conference ADC2003*, Adelaide, Australia, 2003.

[23] Wang H., Cao J., and Zhang Y. An Electronic Payment Scheme and Its RBAC management. *Concurrent Engineering: Research and Application*, 12(3):247–275, 2004.

[24] Wang H., Cao J., Zhang Y. A flexible payment scheme and its role based access control. *IIEEE Transactions on Knowledge and Data Engineering*, 17(3):425–436, 2005.

[25] Wang H., Li J., Addie R., Dekeyser S. and Watson R. A framework for role-based group delegation in distributed environment. In *Proceedings of the 29th Australasian Computer Science Conference*. Australian Computer Society, 2006.

[26] Wang H., Zhang Y., Cao J. and Kambayahsi Y. A global ticket-based access scheme for mobile users. *Special Issue on Object-Oriented Client/Server Internet Environments, Information Systems Frontiers*, 6(1):35–46, 2004.

[27] Wang H., Zhang Y., Cao J. and Varadharajan V. Achieving secure and flexible m-services through tickets. *IEEE Transactions on Systems, Man, and Cybernetics, Part A, Special issue on M-Services*, pages 697–708, 2003.

[28] Zhang L., Ahn G., and Chu B. A role-based delegation framework for healthcare information systems. In *Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, pages 125–134, Monterey, CA, June 2002.

[29] Zhang L. and Ahn G. and Chu B. A rule-based framework for role-based delegation and revocation. *ACM Trans. Inf. Syst. Secur.*, 6(3):404–441, 2003.

[30] Zhao X. and Liu C. Version management in the business process change context. In *5th International Conference on Business Process Management*, volume 4714, pages 198–213. Lecture Notes in Computer Science, Springer, 2007.