The ARBAC99 Model for Administration of Roles

Ravi Sandhu and Qamar Munawer

Laboratory for Information Security Technology (LIST)
George Mason University, ISE Department MS 4A4, Fairfax, VA 22030
sandhu@gmu.edu, www.list.gmu.edu

Abstract

Role-Based Access Control (RBAC) is a flexible and policy-neutral access control technology. For large systems—with hundreds of roles, thousands of users and millions of permissions—managing roles, users, permissions and their interrelationships is a formidable task that cannot realistically be centralized in a small team of security administrators. An appealing possibility is to use RBAC itself to facilitate decentralized administration of RBAC. The ARBAC97 (administrative RBAC '97) model was recently introduced for this purpose. ARBAC97 has three sub-models called URA97 (for user-role administration), PRA97 (for permission-role administration) and RRA97 (for role-role administration).

In this paper we define enhancements to ARBAC97 to give us the new ARBAC99 model. Specifically the URA and PRA sub-models of ARBAC99 introduce significant new features relative to their counterparts in ARBAC97 (while RRA is left unchanged). ARBAC99 incorporates the concept of mobile and immobile users and permissions for the first time in this arena. This paper gives a formal definition of ARBAC99, motivates these enhancements and analyzes several subtle issues that arise in this context.

1 Introduction

Role-based access control (RBAC) is a promising access control technology for the modern computing environment (for recent literature, see [BFA99, GGF98, FBK99, NO99, SCFY96, San98, SBM99, ZSS99]). In RBAC permissions are associated with roles, and users are assigned to appropriate roles thereby acquiring the roles' permissions. This greatly simplifies management. Roles are created for various job functions in an organization and users are assigned roles based on responsibilities and qualifications. Users can be easily reassigned from one role to another. Roles can be granted new permissions as new applications come on line, and permissions can be revoked from roles as needed. Role-role relationships can be established to lay out broad policy objectives.

RBAC is policy neutral and flexible. The policy enforced is a consequence of the detailed configuration of various RBAC components. RBAC allows a wide range of policies to be implemented. Administration of RBAC must be carefully controlled to ensure the policy does not drift away from its original objectives. In large systems the number of roles can be in the hundreds or thousands, users can be in the tens or hundreds of thousands and permissions in the millions. Managing these roles and users, and their interrelationships is a formidable task that cannot realistically be centralized in a small team of security administrators. Decentralizing the details of RBAC administration without loosing central control over broad policy is a challenging goal for system designers and architects. There is tension here between the desire for scalability through decentralization and maintenance of tight control.

Since the main advantage of RBAC is to facilitate administration, it is natural to ask how RBAC itself can be used to manage RBAC. The use of RBAC for managing RBAC will be an important factor in its long-term success. There are many components to RBAC. RBAC administration is therefore multifaceted. In particular we can separate the issues of assigning users to roles, assigning permissions to roles, and assigning roles to roles to define a role hierarchy. These activities are all required to bring users and permissions together. However, in many cases, they are best done by different administrators or administrative roles. Assigning permissions to roles is typically the province of application administrators. Thus a banking application can be implemented so credit and debit operations are assigned to a teller role, whereas approval of a loan is assigned to a managerial role. Assignment of actual individuals to the teller and managerial roles is a personnel management function. Assigning roles to roles has aspects of user-role and permission-role administration. More generally, role-role relationships establish broad policy.

An administrative model called ARBAC97 (administrative RBAC '97) was recently introduced by Sandhu et al [SBM99]. ARBAC97 has three components: URA97 is concerned with user-role administration, PRA97 is concerned with permission-role administration and is a dual of URA97, and RRA97 deals with role-role administration.

In this paper we introduce and formalize a significant enhancement to ARBAC97 to give us the new ARBAC99 model. Changes are specifically made to the URA97 and PRA97 models, while RRA97 remains unchanged. In other words, URA99 and PRA99 are respectively different from URA97 and PRA97 while RRA99 is identical to RRA97. The important difference between ARBAC99 and ARBAC97 is the concept of mobile and immobile users and permissions. We explain and motivate this distinction informally in the next section. Following this intuitive explanation we discuss the URA97 model and its enhancement to URA99. PRA99 is a dual of URA99 and is defined in this paper in this manner rather than being evolved from PRA97. We assume the reader is generally familiar with RBAC concepts. Our underlying RBAC model is the so-called RBAC96 model [SCFY96].

2 Mobility and Immobility

We now informally motivate the intuition behind URA97 and the notion of mobility and immobility of users introduced in URA99. Consider the role hierarchy of figure 1(a) and the administrative role hierarchy of figure 1(b). We will use these hierarchies in our examples throughout this paper. Figure 1(a) shows the regular roles that exist in an engineering department. In these diagrams senior roles shown towards the top inherit permissions from junior roles shown towards the bottom. Equivalently junior roles inherit members from senior roles. Thus permissions assigned to E1 are also available to members of PL1 but not vice versa. Alternately, members of PL1 are also members of E1 but not vice versa.

Figure 1(a) has a junior-most role E to which all employees in the organization belong. Within the engineering department there is a junior-most role ED and senior-most role DIR.[1] In between there are roles for two projects within the department, project 1 on the left and project 2 on the right. Each project has a senior-most project lead role (PL1 and PL2) and a junior-most engineer role (E1 and E2). In between each project has two incomparable roles, production engineer (PE1 and PE2) and quality engineer (QE1 and QE2). The role hierarchy can, of course, be extended to dozens or hundreds of projects within the engineering department. Moreover, each project could have a different structure for its roles. The example can also be extended to multiple departments each with different structure and policies.

Figure 1(b) shows the administrative role hierarchy which co-exists with figure 1(a). The senior-most administrative role is the senior security officer (SSO). The administrative roles junior to SSO consist of two project security officer roles (PSO1 and PSO2) and a department security officer (DSO) role with the relationships illustrated in the figure.[2]

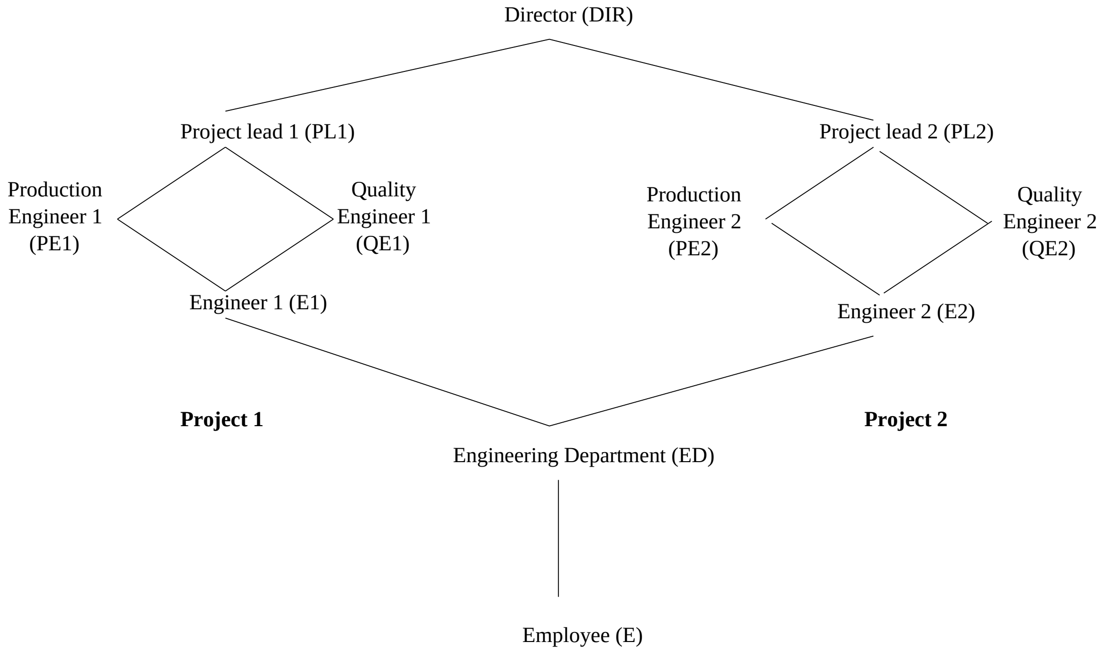
There are two issues that need to be addressed in decentralized administration of user-role membership. Firstly we need to control the roles that an administrative role has authority over. We would like to say, for example, that the PSO1 administrative role controls membership in project 1 roles, i.e., E1, PE1, QE1 and PL1. Secondly, it is also important to control which users are eligible for membership in these roles.

URA97 addresses these two issues respectively by means of a *role range* and a *prerequisite role* (or more generally a *prerequisite condition*). A role range is specified by a junior and senior role. The range includes all roles between these two endpoints. The [and] brackets indicate that respectively the junior and senior end point is included in the range, whereas the (and) brackets indicate the end point is excluded. Thus [E1,PL1] consists of E1, PE1, QE1 and PL1, while [E1,PL1) omits PL1.[3] The prerequisite role specifies which users can be assigned by PSO1 to roles in the authorized range. For example, if a prerequisite role of ED is specified for PSO1 and role range [E1,PL1] then only those users who are already members of ED are eligible to be assigned to a role in [E1,PL1] by the PSO1 role. This simple idea is

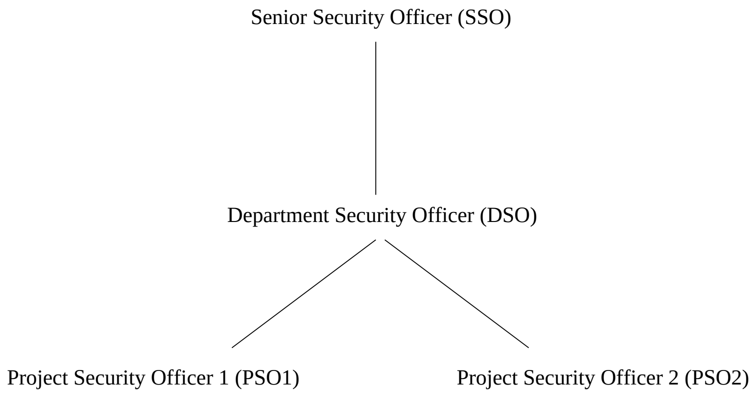
[1] It is not necessary to have a junior-most role or senior-most role in every role hierarchy, so this is just an artifact of our example.

[2] In addition we assume there is a chief security officer who is authorized to make any change in the system, and as such is outside the purview of these administrative models.

[3] The reader may recognize this as standard mathematical notation for open and closed intervals.



(a) Roles



(b) Administrative Roles

Figure 1: Example Role and Administrative Role Hierarchies

surprisingly powerful as demonstrated in [SBM99].

In URA97 there are two consequences of assigning a user to a role. Firstly the user is authorized to use the permissions of that role and its juniors. Secondly, the user also becomes eligible for assignment to other roles by appropriate administrative roles. These two aspects of role membership are tightly and inextricably coupled in URA97. The main innovation in URA99 is to decouple these two aspects.

URA99 distinguishes two kinds of membership in a role. Immobile membership grants the user the authority to use the permissions of a role but does not make that user eligible for further role assignments. Thus immobile members of a role only get the first aspect of role membership identified above. Mobile membership on the other hand covers both aspects.

This distinction between mobile and immobile users can be very useful in practice. For example, a user under training can be assigned to the ED role and thus participate in the engineering department while preventing junior administrators from assigning this user to projects. After completion of training the user's membership in ED can be upgraded to be mobile. Another example is a visitor who can be granted immobile membership in a junior role as an observer but cannot get assigned to more senior roles. Yet another example is a consultant who might be assigned to the E2 role as an immobile member. The consultant can participate in project 2 and use the general resources of the engineering department due to inherited membership in ED. At the same time the immobility of the consultant prevents junior administrators from assigning the consultant to project 1 roles.

There is a dual notion of mobility for permissions introduced in PRA99. PRA97 allows us to give PSO1 the authority to take a permission assigned to PL1 and grant it to roles in the range [E1,PL1). The idea is that each project can delegate permissions of the project lead role to more junior project roles as the project security officers deem appropriate. While this may be acceptable for most permissions of the project lead role it is likely that some permissions are not suitable for such delegation. These permissions can be assigned to PL1 as immobile while the others can be assigned as mobile.

It is possible that this distinction between mobile and immobile users and permissions can be simulated in some way in ARBAC97 without directly incorporating it in the extended model ARBAC99. This would be an interesting theoretical exercise which is beyond the scope of this paper. Our objective in this paper is to understand the semantics of mobility and immobility and its interaction with the role hierarchy. As we will see there are a number of subtle issues that arise in formalizing this intuition.

3 The URA97 Model

In this section we briefly review the formal definition of the URA97 model. Our discussion is necessarily brief and the reader is referred to [SB97, SBM99] for detailed motivation and rationale for the design of URA97. URA97 was introduced by Sandhu and Bhamidipati [SB97] and was later incorporated in ARBAC97 [SBM99]. A number of implementations of URA97 on various platforms have also been reported in the literature [SA98a, SA98b, SP98, SB99].

URA97 has two components, one dealing with assignment of users to roles (the grant model) and the other with revocation of user membership (the revoke model).

3.1 URA97 Grant Model

The notion of a prerequisite condition is a key part of URA97.

Definition 1 Let R be the set of regular roles, U the set of users and $UA \subseteq U \times R$ the user-role assignment relation. A **prerequisite condition** is a boolean expression using the usual $\wedge$ and $\vee$ operators on terms of the form x and $\overline{x}$ where $x \in R$. For a given set of roles R let CR denotes all possible prerequisite conditions that can be formed using the roles in R.

Definition 2 A prerequisite condition is evaluated for a user u by interpreting x to be true if $(\exists x' \geq x)(u, x') \in UA$ and $\overline{x}$ to be true if $(\forall x' \geq x)(u, x') \notin UA$ where $\geq$ is the senior relation in the role hierarchy.

User-role assignment is controlled in URA97 by the *can-assign* relation as follows.

Definition 3 User-role assignments are authorized by the relation, *can-assign* $\subseteq AR \times CR \times 2^R$ where AR is the set of administrative roles and subsets of the roles R are identified using the range notation discussed in section 2. (URA97 requires that$AR \cap R = \emptyset$.)

The meaning of *can-assign*$(x, y, \{a, b, c\})$ is that a member of the administrative role x (or a member of an administrative role senior to x) can assign a

Admin. Role	Prereq. Condition	Role Range
PSO1	ED	[E1, PL1)
PSO2	ED	[E2, PL2)
DSO	ED $\wedge$ $\overline{\text{PL2}}$	[PL1, PL1]
DSO	ED $\wedge$ $\overline{\text{PL1}}$	[PL2, PL2]
SSO	ED	(ED, DIR]
SSO	E	[ED, ED]

Table 1: Example of *can-assign* in URA97

Admin. Role	Role Range
PSO1	[E1, PL1)
PSO2	[E2, PL2)
DSO	(ED, DIR)
SSO	[ED, DIR]

Table 2: Example of *can-revoke* in URA97

user whose current membership, or non-membership, in regular roles satisfies the prerequisite condition y to be a member of regular roles a, b or c.

An example of *can-assign* is given in table 1. PSO1 can assign membership in E1, PE1 or QE1 to members of ED. Similarly, for PSO2 with respect to E2, PE2 and QE2. DSO can assign users in ED to PL1 provided the user is not already a member of PL2. Similarly for PL2 with respect to PL1. SSO can assign members of E to ED and members of ED to any role in the engineering department.

3.2 URA97 Revoke Model

In classical discretionary access control the source (direct or indirect) of a permission and the identity of the revoker is typically taken into account in interpreting the revoke operation. URA97 takes a role-based approach to revocation so authority to revoke is independent of who actually assigned a user to a role, as follows.

Definition 4 The URA97 model authorizes user-role revocation by means of the relation *can-revoke* $\subseteq AR \times 2^R$.

The meaning of *can-revoke*(x, Y) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can revoke membership of a user from any regular role $y \in Y$. Y is specified using the range notation. We say Y defines the *range of revocation*.

An example of *can-revoke* is given in table 2. The role ranges in tables 1 and 2 are closely related. This is likely to be the common case but URA97 does not require that *can-assign* and *can-revoke* have correlated role ranges.

To understand the semantics of revocation we introduce the following distinction.

Definition 5 Let us say a user u is an *explicit member* of role x if $(u, x) \in UA$, and that u is an *implicit member* of role x if for some $x' > x$, $(u, x') \in UA$.

For example, an explicit member of DIR in the engineering department role hierarchy is an implicit member of all other roles. It is possible for a user to simultaneously be an explicit and implicit member of a role.

Revocation in URA97 has impact only on explicit membership and is said to be weak. Thus a user may be revoked explicitly from E1 but continue to be an implicit member due to explicit membership in, say, PL1. Strong revocation requires revocation of both explicit and implicit memberships. So a user who is strongly revoked from E1 will be weakly revoked from E1 and all roles senior to E1. Strong revocation therefore has a cascading effect upwards in the role hierarchy. Of course, each of the weak revokes required for this purpose must be authorized. Strong revocation can be defined to have an all-or-nothing semantics (so no revocation takes place if even one of the required weak revokes fails) or a best-effort semantics (so all required weak revokes within the authorized revocation range take effect, while those outside the authorized range fail). In the formal URA97 model strong revocation is defined to be a series of weak revocations (although in an implementation direct support for strong revocation would be more efficient).

4 The URA99 Model

URA99 builds upon the URA97 model by introducing the following concept, motivated earlier in section 2. A user's membership in a role can be mobile or immobile. *Mobile membership* of user u in role x means that u can use permissions of role x and members of administrative roles can use this membership to put user u into other roles. *Immobile membership* of user u in role x means that u can use permissions of role x but members of administrative roles cannot use this membership to put user u into other roles.

To formalize this distinction we consider each role x as consisting of two sub-roles Mx and IMx. Membership in Mx is mobile whereas membership in IMx is immobile. For compatibility with URA97 we define the set of roles R to consist of the mobile and immobile sub-roles as follows.

Definition 6 For a given set of roles $R1$ we define the roles for URA99 to be $R = \{Mx, IMx \mid x \in R1\}$.

With this definition the user-assignment relation of URA97, $UA \subseteq U \times R$ essentially remains unchanged in URA99. Assignment of a user to Mx signifies that the user is a mobile member of x. Similarly, assignment of a user to IMx signifies that the user is an immobile member of x.

Combining mobile and immobile membership with the previously defined notion of explicit and implicit membership gives us four distinct kinds of role membership in URA99, as follows.

Definition 7 There are four kinds of user-role membership in URA99 for any given role x.

- *Explicit Mobile Member EMx*
  $u \in EMx \equiv (u, Mx) \in UA$
- *Explicit Immobile Member EIMx*
  $u \in EIMx \equiv (u, IMx) \in UA$
- *Implicit Mobile Member ImMx*
  $u \in ImMx \equiv (\exists x' > x)(u, Mx') \in UA$
- *Implicit Immobile Member ImIMx*
  $u \in ImIMx \equiv (\exists x' > x)(u, IMx') \in UA$

It is possible for a user to have all four kinds of membership in a role at the same time. However, we will define the semantics of URA99 so that there is strict precedence amongst these four kinds of membership as follows.

$$EMx > EIMx > ImMx > ImIMx$$

So even though a user can have multiple kinds of membership in a role, at any time only one of those is actually in effect.

To explain the inheritance of mobile and immobile memberships of a role we first consider the hierarchy of two roles shown in figure 2(a) where role $x1$ is senior to role $x2$. User Alice who is an explicit mobile member of role $x1$ (Alice $\in EMx1$) is an implicit mobile member of $x2$ (Alice $\in ImMx2$). Similar inheritance applies to immobile memberships as well. Therefore, if Bob is an explicit immobile member of role $x1$ (Bob $\in EIMx1$) he is also an implicit immobile member of role $x2$ (Bob $\in ImIMx2$).

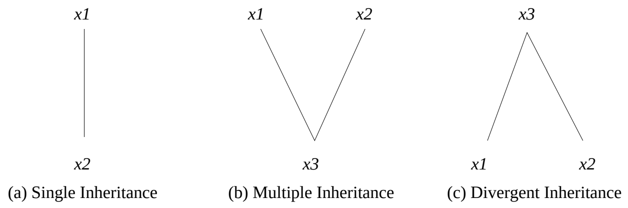


Figure 2: Inheritance of mobility and immobility

Next, consider the role hierarchy of figure 2(b). Let Bob be an explicit mobile member of role $x1$ and explicit immobile member of role $x2$. Now Bob is an implicit mobile member of $x3$ (because of his explicit mobile membership in $x1$) and is an implicit immobile member of role $x3$ (because of his explicit immobile membership in $x2$). According to the precedence rule mobile membership is stronger than immobile membership. This means that Bob will effectively have implicit mobile membership in role $x3$.

Finally consider the role hierarchy of figure 2(c). Say that Bob is an explicit mobile member of role $x3$. Therefore Bob is an implicit mobile member of roles $x1$ and $x2$ in the hierarchy. Let us also suppose that Bob is an explicit immobile member of $x2$. Then according to our precedence rule Bob will be immobile in $x2$ whereas mobile in $x1$. Bob's explicit immobile membership overrules implicit mobile membership.

The meaning of a prerequisite condition in URA97 is quite straightforward, because the notion of role membership is simple. In URA99 we need to interpret a prerequisite condition in terms of mobile-immobile and explicit-implicit memberships. URA99 prerequisite conditions are defined in terms of x and $\overline{x}$ as in URA97 (rather than in terms of Mx, IMx, $\overline{Mx}$ and $\overline{IMx}$). Membership and non-membership in a role is then interpreted as follows.

Definition 8 In the URA99 grant model a prerequisite condition is evaluated for a user u by interpreting x to be true if

$$u \in EMx \vee (u \in ImMx \wedge u \notin EIMx)$$

and $\overline{x}$ to be true if

Admin. Role	Prereq. Role	Role Range
PSO1	ED	[E1, PL1)
PSO2	ED	[E2, PL2)
DSO	ED $\wedge$ $\overline{\text{PL2}}$	[PL1, PL1]
DSO	ED $\wedge$ $\overline{\text{PL1}}$	[PL2, PL2]
SSO	ED	(ED, DIR]
SSO	E	[ED, ED]

Table 3: Example of *can-assign-M*

Admin. Role	Prereq. Role	Role Range
PSO1	ED	[E1, PL1)
PSO2	ED	[E2, PL2)
DSO	ED $\wedge$ $\overline{\text{PL2}}$	[PL1, PL1]
DSO	ED $\wedge$ $\overline{\text{PL1}}$	[PL2, PL2]
SSO	ED	(ED, DIR]
SSO	E	[ED, ED]
DSO	E	[ED, ED]

Table 4: Example of *can-assign-IM*

$$u \notin EMx \wedge u \notin EIMx \wedge u \notin ImMx \wedge u \notin ImIMx$$

In other words x denotes mobile membership (explicit or implicit) and $\overline{x}$ denotes absence of any kind of membership. Note that it is not possible for x and $\overline{x}$ to be simultaneously true. They can however be simultaneously false (when $u \in EIMx$ and $u \notin EMx$).

The *can-assign* relation of URA97 is replaced by two relations as follows.

Definition 9 User-role assignments as mobile members are authorized by the relation, *can-assign-M* $\subseteq AR \times CR \times 2^R$ and user-role assignments as immobile members are authorized by the relation, *can-assign-IM* $\subseteq AR \times CR \times 2^R$.

The meaning of *can-assign-M(x, y, Z)* is that a member of administrative role x (or a member of administrative role senior to x) can assign a user whose current membership, or non-membership, in regular roles satisfies the prerequisite y to a regular role $z \in Z$ as a mobile member. Whereas the meaning of *can-assign-IM(x, y, Z)* is that a member of administrative role x (or a member of administrative role senior to x) can assign a user whose current membership, or non-membership, in regular roles satisfies the prerequisite y to a regular $z \in Z$ as an immobile member.

Examples of *can-assign-M* and *can-assign-IM* are respectively shown in tables 3 and 4. Table 3 is identical to table 1. Of course, the prerequisite condition is now interpreted differently as discussed above. The top six rows of table 4 are identical to table 3. This means that mobile or immobile membership is granted at discretion of the individual administrators. URA99 requires that authorization for granting mobile or immobile membership be explicitly specified in this manner. There is no implication in general that authority to grant mobile membership implies authority to grant immobile membership (although this may be a common case). The last row of table 4 authorizes DSO to enroll any employee as an immobile member of ED. DSO does not have the power to enroll employees as mobile members of ED. That power is confined to SSO. In this example DSO can enroll an employee as a immobile member of ED and later the SSO can upgrade the membership to be mobile.

4.1 URA99 Revoke Model

The URA99 revoke model fixes a lack of symmetry between the grant and revoke models of URA97 which is quite independent of the issue of mobility. It also deals with revocation of mobile and immobile membership.

In URA97 the relation *can-assign* involves the prerequisite conditions but *can-revoke* does not. To see the utility of prerequisite conditions in this context consider that PSO1 controls user role assignments in project 1 roles. If Bob is a member of E1 then PSO1 can assign Bob to any role of project 1, namely E1, PE1, and QE1. These assignments are governed by the relation *can-assign*. Suppose PSO1 does not want Bob to be a member of any role outside project 1 because his membership in other roles may affect his performance in project 1. If Bob is assigned to a role that falls outside project 1 roles then PSO1 should have authority to revoke him from that role. URA97 does not provide this conditional authority to revoke a role. Prerequisite conditions in *can-revoke* are one means to provide this facility.

Following the approach of the grant model in URA99 we introduce two relations to authorize revocation of mobile and immobile membership as follows.

Definition 10 The URA99 model authorizes revocation of mobile membership by the relation *can-revoke-M* $\subseteq AR \times CR \times 2^R$ and revocation of immobile membership by the relation *can-revoke-IM* $\subseteq AR \times CR \times 2^R$.

Admin. Role	Prereq. Role	Role Range
PSO1	E	[E1, PL1)
PSO2	E	[E2, PL2)
DSO	E	(ED, DIR)
SSO	E	[ED, DIR]
PSO1	E1	[E2, PL2)
PSO2	E2	[E1, PL1)

Table 5: Example of *can-revoke-M*

Admin. Role	Prereq. Role	Role Range
PSO1	E	[E1, PL1)
PSO2	E	[E2, PL2)
DSO	E	(ED, DIR)
SSO	E	[ED, DIR]
PSO1	E1	[E2, PL2)
PSO2	E2	[E1, PL1)
DSO	E	[ED, ED]

Table 6: Example of *can-revoke-IM*

The meaning of *can-revoke-M(x, y,* $\{a, b, c\}$*)* is that a member of administrative role x (or a member of a administrative role senior to x) can revoke mobile membership of a user from role a, b or c subject to the prerequisite condition y. Similarly for *can-revoke-IM* with respect to immobile membership.

An example of these relations is given in tables 5 and 6. The first four rows of tables 5 are essentially the same as table 2. The prerequisite condition E will evaluate to true for all employees so is redundant. It can equivalently be replaced by any predicate that is always true. The last two rows allow PSO1 to revoke project 1 users from project 2 roles and vice versa. Th top six rows of table 6 are identical to table 5, so in this example authority to revoke mobile membership also implies authority to revoke immobile membership. The last row of table 6 authorizes DSO to remove immobile users from ED (which DSO has the power to assign as per table 4).

The evaluation of a prerequisite condition for the revoke model of URA99 is different from the grant model. For the revoke model we do not distinguish mobile and immobile membership. Thus we have the following interpretation.

Definition 11 In the URA99 revoke model a prerequisite condition is evaluated for a user u by interpreting x to be true if

$$u \in EMx \vee u \in EIMx \vee u \in ImMx \vee u \in ImIMx$$

and $\overline{x}$ to be true if

$$u \notin EMx \wedge u \notin EIMx \wedge u \notin ImMx \wedge u \notin ImIMx$$

Note that unlike in the grant model x and $\overline{x}$ cannot be false at the same time. As in URA97 they are logical complements of each other.

4.2 URA97 as a Special Case of URA99

If all membership is restricted to being mobile than URA99 is identical to URA97. This can be achieved by setting *can-assign-IM* and *can-revoke-IM* to be empty. In this manner there is a simple relationship between URA97 and URA99.

5 The PRA99 Model

The PRA99 model deals with assignment and revocation of permissions to and from the roles. Like users, permissions can also be assigned to roles as mobile and immobile. Just as PRA97 relates to URA97, we have PRA99 as an exact dual of URA99. For sake of completeness we give a complete definition of PRA99 here. The main difference between PRA99 and URA99 is that in PRA99 implicit membership of a permission in a role is inherited upwards in the hierarchy. We give the definitions below without further commentary since they are so closely related to URA99 definitions.

Definition 12 The roles in PRA99 are the same as in URA99, that is, $R = \{Mx, IMx \mid x \in R1\}$. The permission role assignment relation is $PA \subseteq U \times R$.

Definition 13 There are four kinds of permission-role membership in PRA99 for any given role x.

- *Explicit Mobile Member EMx*
  $p \in EMx \equiv (p, Mx) \in PA$
- *Explicit Immobile Member EIMx*
  $p \in EIMx \equiv (p, IMx) \in PA$
- *Implicit Mobile Member ImMx*
  $p \in ImMx \equiv (\exists x' < x)(p, Mx') \in PA$
- *Implicit Immobile Member ImIMx*
  $p \in ImIMx \equiv (\exists x' < x)(p, IMx') \in PA$

Definition 14 Permission-role assignments as mobile members are authorized by the relation, *can-assignp-M* $\subseteq AR \times CR \times 2^R$ and permission-role assignments as immobile members are authorized by the relation, *can-assignp-IM* $\subseteq AR \times CR \times 2^R$.

Definition 15 The PRA99 model authorizes revocation of mobile membership by the relation *can-revokep-M* $\subseteq AR \times CR \times 2^R$ and revocation of immobile membership by the relation *can-revokep-IM* $\subseteq AR \times CR \times 2^R$.

Definition 16 In the PRA99 grant model a prerequisite condition is evaluated for a permission p by interpreting x to be true if

$$p \in EMx \vee (p \in ImMx \wedge p \notin EIMx)$$

and $\overline{x}$ to be true if

$$p \notin EMx \wedge p \notin EIMx \wedge p \notin ImMx \wedge p \notin ImIMx$$

Definition 17 In the PRA99 revoke model a prerequisite condition is evaluated for a permission p by interpreting x to be true if

$$p \in EMx \vee p \in EIMx \vee p \in ImMx \vee p \in ImIMx$$

and $\overline{x}$ to be true if

$$p \notin EMx \wedge p \notin EIMx \wedge p \notin ImMx \wedge p \notin ImIMx$$

6 Discussion and Conclusion

We have described the new ARBAC99 model for role-based administration of role-based access control. ARBAC99 is the first model that incorporates the notion of mobile and immobile users and permissions in administrative RBAC. It has three components: URA99 (user-role administration '99), PRA99 (permission-role administration '99) and RRA99 (role-role administration '99). It is an extension of ARBAC97 obtained by adding the concept of mobile users and permissions in the URA and PRA models. The RRA model is unchanged.

The basic intuition of ARBAC97 is not altered in this paper, that is, the decentralization of administration of user-role assignments, permission-role assignments and role-role hierarchies by means of administrative roles, prerequisite conditions and role ranges. Administrative roles are given autonomy within their administrative ranges as constrained by prerequisite conditions.

References

[BFA99] Elisa Bertino, Elena Ferrari, and Vijay Atluri. Specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security*, 2(1), February 1999.

[FBK99] David F. Ferraiolo, John F. Barkley, and D. Richard Kuhn. A role based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security*, 2(1), February 1999.

[GGF98] Virgil D. Gligor, Serban I. Gavrila, and David Ferraiolo. On the formal definition of separation-of-duty policies and their composition. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pages 172–183, Oakland, CA, May 1998.

[NO99] Matunda Nyanchama and Sylvia Osborn. The role graph model and conflict of interest. *ACM Transactions on Information and System Security*, 2(1), February 1999.

[SA98a] Ravi Sandhu and Gail-Joon Ahn. Decentralized group hieraches in unix: An experiment and lessons learned. In *Proceedings of 21st NIST-NCSC National Information Systems Security Conference*, Arlington, VA, October 5-8 1998.

[SA98b] Ravi Sandhu and Gail-Joon Ahn. Group hierarchies with decentralized user assignment in Windows NT. In *Proc. International Association of Science and Technology for Development (IASTED) Conference on Software Engineering*, Las Vegas, Nevada, October 1998.

[San98] Ravi Sandhu. Role-based access control. In Zelkowitz, editor, *Advances in Computers, Volume: 46*. Academic Press, 1998.

[SB97] Ravi Sandhu and Venkata Bhamidipati. The URA97 model for role-based administration of user-role assignment. In T. Y. Lin and Xiaolei Qian, editors, *Database Security XI: Status and Prospects*. North-Holland, 1997.

[SB99] Ravi S. Sandhu and Venkata Bhamidipati. Role-based administration of user-role assignment: The URA97 model and its Oracle implementation. *The Journal Of Computer Security*, 1999. in press.

[SBM99] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security*, 2(1), February 1999.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[SP98] Ravi Sandhu and Joon Park. Decentralized user-role assignment for web-based intranets. In *Proceedings of 3rd ACM Workshop on Role-Based Access Control*, pages 1–12, Fairfax, VA, October 22-23 1998. ACM.

[ZSS99] M. Zurko, R. Simon, and T. Sanfilippo. A user-centered modular authorization service built on an rbac foundation. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pages 57–71, Oakland, CA, May 1999.