Specifying Role-Based Access Constraints With Object Constraint Language

Hua Wang[1], Yanchun Zhang[2], Jinli Cao[3], and Jian Yang[4]

[1] Department of Maths & Computing, University of Southern Queensland
Toowoomba QLD 4350 Australia
wang@usq.edu.au
[2] School of Computer Science and Mathematics
Victoria University of Technology, Melbourne City, MC8001, Australia.
yzhang@csm.vu.edu.au
[3] Department of Computer Science & Computer Engineering
La Trobe University, Melbourne, VIC 3086, Australia
jinli@cs.latrobe.edu.au
[4] School of Information Technology
Swinbourne University of Technology, Hawthorn, Victoria, 3122, Australia
jyang@it.swin.edu.au

Abstract. Constraints are an important aspect of role-based access control (RBAC). Constraints have to be satisfied when an administrator wants to assign (revoke) a role to a user or a permission to a role. The importance of constraints associated with user-role assignments and permission-role assignments in RBAC has been recognized but the modelling of these constraints has not been received much attention. In this paper we use a de facto constraints specification language in software engineering to analyze the constraints in user-role assignments and permission-role assignments. We show how to represent role-based access constraints with object constraint language (OCL) and discuss the future work. Finally, comparisons with other related work are presented.

1 Introduction

The National Institute of Standards and Technology (NIST) developed role-based access control (RBAC) prototype in 1995 [6] and published a formal model [8]. 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. As shown in Fig. 1, the relationships between users and roles, and between roles and permissions are many-to-many. Roles are in two categories within RBAC, one is administrative roles (admin.role), the other is regular roles (role) that need to be assigned to or revoked from users by administrative roles. The relationship of Senior-Junior shows hierarchies between roles. Senior roles inherit permissions from

junior roles. Let $x > y$ denote x is senior to y with obvious extension to $x \geq y$. Constraints exist within user-role assignment (UA), permission-role assignment (PA) and Senior-Junior relationships etc.

There are various constraints in RBAC as shown in Fig. 1. Constraints can be described in natural languages, such as English, German or in other formal languages. Practice has shown that constraints in natural languages will always result in ambiguities [2]. Constraints in formal language are usable to persons with a strong mathematical background, but difficult for average business or system developers to use. For instance, Ahn and Sandhu [3] proposed a formal language called RCL99 (Role-based constraints specification language 1999) and identified authorization constraints such as prohibition and obligation constraints. The authors of RCL99 are security experts and system developers may not understand the constraints language before they accept training of roles, users and permissions etc in RBAC. This paper focuses on constraints specification, that is how constraints can be represented.

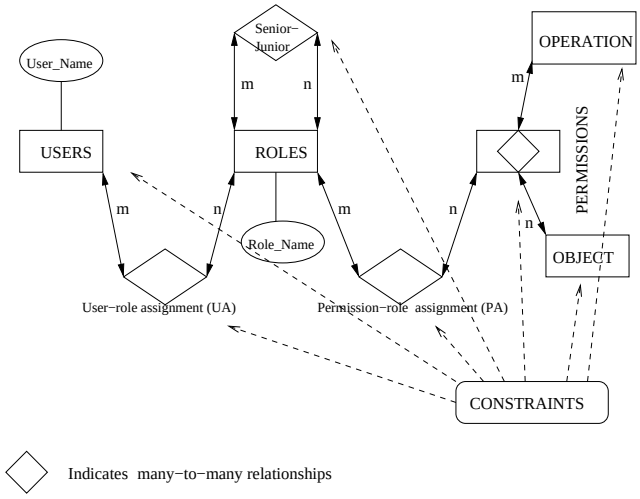

Fig. 1. RBAC relationship

The paper is organized as follows. In the next section, we identify the motivation of our work in this paper and review the related technologies such as RBAC, Unified Modelling Language (UML) and OCL. Constraints in user-role assignment and permission-role assignment are discussed in section 3. The identified constraints such as Dynamic Separation of Duty (DSD), Static Separation of Duty (SSD) and Operation authorization are expressed with OCL in section 4. Comparisons with others' work are analyzed in section 5 and the conclusions are in section 6.

2 Motivation and related technologies

RBAC constraints are one of the most important components that have involved in the principal motivations of RBAC analysis and design. These constraints described by mathematical expressions are difficult to understand and use for business people and system developers. OCL, as an industrial standard constraints specification language, has been used to describe constraints in system analysis and design. Approaches represented by OCL are easy for system developers and modelers to specify and to understand the constraints in different kinds of RBAC management.

2.1 Role-based access control

The RBAC security model has two components: MC_0 and MC_1. Model component MC_0, called the RBAC authorization database model, defines the RBAC security properties for authorization of static roles. Static properties of a RBAC authorization database include role hierarchy, inheritance, cardinality, and static separation of duty. MC_1 called the RBAC activation model, defines the RBAC security properties for dynamic activation of roles. Dynamic properties include role activation, permission execution, dynamic separation of duties, and object access. In particular, the RBAC model supports the specification of:

a. User/role associations; the constraints specifying user authorizations to perform roles,

b. Permission/role associations; the constraints identifying role authorizations to run permissions,

c. Role hierarchies; the constraints specifying which role may inherit all of the permissions of another role,

d. Duty separation constraints; these are role/role associations indicating conflict of interest:

d1. Static separated duty (SSD); a constraint specifying that a user cannot be authorized for two different roles,

d2. Dynamic separated duty (DSD); a constraint specifying that a user can be authorized for two different roles but cannot act simultaneously in both,

f. Cardinality; the maximum number of users allowed, i.e. how many users can be authorized for any particular role (role cardinality), e.g., only one manager.

2.2 Unified Modelling Language and Object Constraints Language

The UML is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex process of system analysis and design and further software implementation. The UML has become a standard modelling language in the field of software engineering [**?**].

Expressions with OCL are described with the context of an instance of a specific type. In an OCL expression, the word **self** is used to indicate the contextual

instance. The keyword **context** is used to represent the type of the context instance of an OCL expression. The label **invar** means that the constraint is an invariant constraint. For example, suppose that students study in a University and they register in a course. These relationships can be modelled with the class model of the UML. If the context is University, then *self* refers to an instance of University. The following shows an example of OCL constraint expression describing a University that has more than 20000 students:

context University **invar**:
self.student $\rightarrow$ size $>$ 20000

The self.student is a set of students that is selected by navigating from University class to Student class though an association. The "." stands for a navigation. A property of a set is accessed by using an arrow "$\rightarrow$ " followed by the name of the property. A property of the set of students is expressed using a keyword "size" in the example. More examples can be reviewed in [11].

3 Constraints in RBAC

Constraints identification and representation are important issues when assigning a role to a user and a permission to a role. This issue has received little attention in the research literature. Most prior work has focused on separation of duty constraints. For example, Chen and Sandhu [5] suggested how constraints can be specified. It may work for a Database Administrator (DBA) or Security Officer to specify constraints. The basic idea to apply constraints is to lay out higher level organizational policy. It does not provide details for the applications of other components in RBAC such as user-role assignment and permission-role assignment. Wang, Cao and Zhang [16] introduced a formal authorization allocation approach and identified the major classes of constraints in RBAC such as *Separation Constraints*, *Prerequisite Constraint* and *Cardinality Constraints*.

3.1 Separation Constraints

There are two kinds of separation constraints. One is Static Separation of Duty while the other one is Dynamic Separation of Duty.

The policy of Static Separation of Duty (SSD) can be centrally specified and can then be uniformly imposed on specific roles. The principle reason with SSD is that a user being authorized as a member of one role, the user cannot be authorized as a member of a second role.

Static Separation of Duty: A user is authorized as a member of a role only if that role is not mutually exclusive with any of the other roles for which the user already possesses membership.

Dynamic Separation of Duty: A user can become a member of a new role only if the proposed role is not mutually exclusive with any of the roles in which the user is already a member. Suppose u is a user and $r_i, r_j (i \neq j)$ are roles, we have:

$\forall u, r_i, r_j$:

(r_i is a member of u) $\wedge$ (r_j is a member of u) $\Longrightarrow$ $r_i \notin$ {mutually-exclusive-activation(r_j) }.

3.2 Prerequisite Constraints

The Prerequisite constraints in user-role assignment can be expressed as below: **Prerequisite condition** is an expression using Boolean operators $\wedge$ and $\vee$ on terms x and $\bar{x}$ where x is a role and $\wedge$ means "and", $\vee$ means "or". 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 $\bar{x}$ to be true if $(\forall x' \geq x), (u, x') \notin UA$, where UA is a set of user-role assignments. $\diamond$

Where $(\exists x' \geq x), (u, x') \in UA$ means that there exists a role x', x' is senior or equal to role x, $(\forall x' \geq x), (u, x') \notin UA$ means that there is no role x', x' is senior or equal to role x, u is a member of role x'.

We use OCL to express prerequisite constraints in the next section.

3.3 Cardinality Constraints

The Cardinality Constraints are a numerical limitation that exist when granting roles to user in user-role assignment. Some roles can only be occupied by a certain number of employees at any given period of time.

3.4 Mobility Constraints

The mobility of user-role relationship is a new feature in user-role assignments. When an administrative role assigns a role to a user with a mobile membership, this allows the user to use the permissions of the role and to be further assigned other roles by administrators. Immobile membership grants the user the authority to use the permissions, but does not make the user eligible for further role assignment. This distinction between mobile and immobile memberships can be very important in practice [10, 14].

4 Constraints expression with OCL

4.1 Separation of duty constraints

Separation of duty is a security principle used to formulate multi-person control policies, requiring that two or more different people be responsible for the completion of a task. The objective of separation of duty is to discourage fraud by spreading the responsibility and authority for an action or task over multiple people. There are two kinds of separation duty. The first one is dynamic separation of duty while the other one is static separation of duty. The objective behind Dynamic Separation of Duty is to allow more flexibility in operations.

Dynamic Separation of Duty places constraints on the simultaneous activation of roles. We now specify these constraints with OCL.

context User **invar:**
let M: Set = { { mutually-exclusive-activation(r: roles)}, ... } **in**
M $\rightarrow$ select($m \mid self.role \rightarrow$ intersection (m) $\rightarrow size >1) \rightarrow$ *is Empty*

This constraint expression selects all mutually exclusive sets, checks all roles assigned to each user, and enforces above requirements.

Static Separation of Duty means that by virtue of a user being authorized as a member of one role, the user is not authorized as a member of a second role. The SSD constraints are expressed by OCL as below.

context User **invar:**
let M: Set = { { mutually-exclusive-authorization(r: roles)}, ... } **in**
M$\rightarrow$ select($m \mid self.role \rightarrow$ intersection (m) $\rightarrow size > 1) \rightarrow$ *is Empty*

This constraint expresses that no two roles in the set of a mutually exclusive authorization can be assigned to one user. The OCL representation for object-based separation of duty is:

context User **invar:**
let M: Set = { { mutually-exclusive-authorization(r: roles)}, ... } **in**
M$\rightarrow$ select($m \mid m \rightarrow$intersection (self.session.object) $\rightarrow size > 1) \rightarrow$ *is Empty*

This constraint indicates that no object associated mutually exclusive roles can be acted by any user at the same time.

4.2 Mobility constraints

It may have hierarchies between roles. We use administrative roles in Fig. 2 to explain the mobility of user-role membership. The Figure shows hierarchies of administrative roles. The administrative role SAR is senior to role AR ($SAR > AR$) (or AR is junior to SAR), and role AR is senior to role JAR ($AR > JAR$) (or JAR is junior to AR). The administrative role SAR inherits administrative role AR with all permissions of AR. Similarly, AR inherits junior administrative role JAR. Each role r is separated into two sub-roles Mr and IMr. Membership in Mr is mobile while membership in IMr is immobile. Assignment of Mr to a user specifies that the user has a mobile membership of r. Similarly, assignment of IMr to a user specifies that the role x is an immobile member of u.

The explicit members of a role r is the set of users $\{u|(u,r) \in UA\}$ and the implicit members of role r is the set of users $\{u|\exists r' > r, (u,r') \in UA\}$ where UA is user-role assignment. Based on the mobile and immobile memberships with the notion of explicit and implicit memberships, there are four kinds of user-role memberships for a given role r [14].

1. *Explicit Mobile Member EMr:*
   $EMr = \{u, |(u, Mr) \in UA\}$
   EMr is a set of users that have mobile membership with role r.

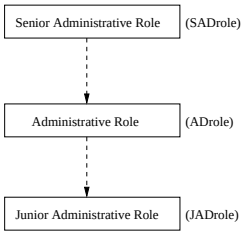

Fig. 2. Hierarchies of administrative roles

2. *Explicit Immobile Member EIMr:*
   $EIMr = \{u, |(u, IMr) \in UA\}$
   $EIMr$ is a set of users that have immobile membership with role r.
3. *Implicit Mobile Member ImMr:*
   $ImMr = \{u, |\exists r' > r, (u, Mr') \in UA\}$
   $ImMr$ is a set of users that have implicit mobile membership with role r. It means that role r has a senior role r', the user in $ImMr$ is a mobile member of r'.
4. *Implicit Immobile Member ImIMr:*
   $ImIMr = \{u, |\exists r' > r, (u, IMr') \in UA\}$
   $ImIMr$ is a set of users that have implicit immobile membership with role r. It means that role r has a senior role r', the user in $ImMr$ is a immobile member of r'.

A user may have all four kinds of membership in a role at the same time. However, we limit strict precedence amongst these four kinds of membership as follows:

$$EMr > EIMr > ImMr > ImIMr$$

Therefore only one of the membership is actually in effect at any time even though a user has multiple kinds of membership in a role. By the definition of the mobile membership, the r has a mobile membership with a user when the user is in EMr or in $ImMr$. However, the role is not a mobile role of a user if the user belongs to both $ImMr$ and $EIMr$ because the strict precedence listed above. Therefore, the Mobility constraints of a user that has a mobile membership of roles can be specified as below:

context Mobile Role **invar:**

self.user $\rightarrow$ select $\{\{u|u \in EMr\} \vee \{\{u|u \in ImMr\} \wedge \{u|u \notin EIMr\}\} \rightarrow$ size ≥ 1

Where $\{u|u \in EMr\}$ is a set of users with a mobile membership of r, and $\{\{u|u \in ImMr\} \wedge \{u|u \notin EIMr\}\}$ is a set of users with an implicit mobile membership of r but not an explicit immobile member of r.

Similarly, since the strict precedence, we have the following expression for immobile membership.

context Immobile Role **invar:**

self.user $\rightarrow$ select $\{\{u|u \in EIMr\} \vee \{\{u|u \in ImIMr\} \wedge \{u|u \notin ImMr\}\} \rightarrow$ size $\geq$1.

4.3 Prerequisite constraints

The following example shows that OCL can also specify prerequisite constraints. Suppose a senior role set of a role r is indicated by *Seniorset (r)*.

$Seniorset(r) = \{r' \mid r'$ *is senior to* $r\ \}$

The *Seniorset (r)* is used for the judgement of whether or not prerequisite conditions are satisfied. For instance, the prerequisite condition of assigning a role r_1 to user u is that the user is already a member of role r_2. Including inheritance, the constraint can be specified as follows:

context User **invar:**

self.role $\rightarrow$ includes(r_1) *implies* (self.role $\rightarrow$ includes (r_2)) $\vee$(self.role $\rightarrow$ includes ($r|r \in Seniorset(r_2)$))

The (self.role $\rightarrow$ includes (r_2)) means that the user is a member of role r_2 while the (self.role $\rightarrow$ includes ($r|r \in Seniorset(r_2)$)) means that the user is a member of a role in the role set $Seniorset(r_2)$.

This constraint is used for user-role assignment (UA). For permission-role assignment (PA), a prerequisite constraint means that a permission p can be assigned to a role only if the role already possesses permission q. This constraint on PA can be specified with OCL expression as follows:

context Permission **invar:**

self.role $\rightarrow$ includes('read exam result ') *implies* self.role $\rightarrow$ includes ('access course')

4.4 Cardinality constraints

Another important constraint in RBAC is a numerical limitation for classes. For example, there is only one person in the role of the head of a department. The head role should be assigned to only one user. The OCL expression for this constraint on UA is as below:

context User **invar:**

self.role $\rightarrow$ select $\{r \mid$ self.name = 'head role' $\} \rightarrow$ size =1

We have represented separation of duty constraints, mobility constraints etc in RBAC by using object constraint language. RBAC is a powerful Web data management system with various constraints. The future work is to prove whether all constraints in RBAC can expressed by OCL or not.

5 Comparisons

There is no much work related to role-based access control models with OCL. The paper of Gail and Michael [2] is the only one that we have found which intro-

duced role-based authorization constraints specification using object constraint language.

The authors in [2] have demonstrated that how to specify role-based authorization constraints using an industrial standard constraints specification language OCL. They specified separation of duty constraints, prerequisite constraints and cardinality constraints and then as a result, utilized constraints identified by a formal language such as role-based constraints specification language (RCL2000) [1] when they designed and analyzed role-based systems. The work in [2] helps system developers to understand constraints and requirements on secure systems development. However, our work substantially differs from that proposal in three aspects. First, our paper has significantly extended the work in [2]. The paper [2] introduced role-based constraints with OCL based on RBAC96 model. RBAC96 model is the first general model for role-based access control that does not even include new conceptions such as mobility etc. By contrast, we deeply discuss various constraint cases in the advanced RBAC model, ARBAC99 [14]. We have provided OCL with not only separation duty constraints, prerequisite conditions but also mobility of user-role assignment and permission-role assignment. Further more, the results in [2] depend on a special model and examples. Second, the discussion area in this paper is much wider than that in paper [2]. The authors in paper [2] focus on role-based authorization constraints. We have discussed details for how to express constraints in RBAC using OCL. The authorization constraints are a part of constraints only in RBAC. For example, the object-based separation of duty is a constraint in RBAC but not an authorization constraint. Third, the OCL expression in our paper is more general. The examples used in [2] represented OCL for role-based authorization constraints are special and hence there is no formal expression. By contrast, we present a number of constraints for role-based access control which allows administrators to authorize a role to users as mobile and immobile member or revoke them from users. The OCL expressions in this paper provide a rich variety of options that can deal the document of administrative roles with regular roles as mobile and immobile members and the expressions have no limits on special examples.

6 Conclusions

This paper has discussed the constraints in RBAC and provided various kinds of constraints representation with object constraint language. We have analysed the constraints in RBAC such as object-based dynamic separation of duty constraints and mobility constraints etc. The constraints including static separation of duty constraints, role-based dynamic and object-based dynamic separation of duty constraints, prerequisite constraints and mobility constraints in user-role assignment and permission-role assignment have been specified using OCL. The work in this paper has significantly extended previous work in several aspects. Namely, the object-based dynamic separation of duty and mobility constraints

with OCL. As a result, we can use object constraints language to represent the constraints in RBAC when we design and analyze role-based access management.

References

1. Ahn G. and Sandhu R.: Role-based authorization constraints specification. Information and System Security, Vol. 3 (4) (2000) 207–226
2. Ahn G. and Shin M.: Role-Based Authorization Constraints Specification Using Object Constraint Language. Tenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (2001) 157 – 165
3. Ahn G. J. and Sandhu R.: The RSL99 Language for Role-Based Separation of Duty Constraints. 4th ACM Workshop on Role-Based Access Control. Fairfax, VA (1999) 43–54
4. Bertino E., Castano S., Ferrari E. and Mesiti M.: Specifying and enforcing access control policies for XML document sources. World Wide Web, 3. Baltzer Science Publishers BV (2000) 139–151
5. Chen F. and Sandhu R.: Constraints for Role-Based Access Control. First ACM Workshop on Role-Based Access Control (1995) 39–46
6. Feinstein H. L.: Final report: NIST small business innovative research (SBIR) grant: role based access control: phase 1. Technical report. SETA (1995)
7. Ferraiolo D., Cugini J. and Kuhn R.: Role-based Access Control (RBAC): Features and motivations. The 11th Annual Computer Security Applications Conference. New Orleans, LA (1995) 241–248
8. Ferraiolo D. F. and Kuhn D. R.: Role based access control. 15th National Computer Security Conference (1992) 554–563
9. Goldschlag D., Reed M., and Syverson P.: Onion routing for anonymous and private Internet connections. Communications of the ACM. Vol. 24 (1999) 39 – 41
10. Oh S. and Sandhu R.: A model for role administration using organization structure. Seventh ACM Symposium on Access Control Models and Technologies. Monterey, California, USA. ACM Press (2002) 155–162
11. Richters M. and Gogolla M.: On Formalizing the UML Object Constraint Language OCL. In Tok-Wang Ling etc editor: 17th International Conference on Conceptual Modeling (ER). Vol. 1507 Springer-Verlag (1998) 449–464
12. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. Addison-Wesley publisher. MA, USA (1999)
13. Sandhu R.: Role-Based Access Control. Advances in Computers. Vol. 46 (1998)
14. Sandhu R. and Munawer Q.: The ARBAC99 model for administration of roles. The Annual Computer Security Applications Conference. Monterey, California, USA. ACM Press (1999) 229–238
15. Wang H., Cao J. and Zhang Y.: A consumer anonymity scalable payment scheme with role based access control. 2nd International Conference on Web Information Systems Engineering (WISE01). Kyoto, Japan (2001) 53–62
16. Wang H., Cao J. and Zhang Y.: Formal Authorization allocation approaches for role-based access control based on relational algebra operations. 3nd International Conference on Web Information Systems Engineering (WISE02). Singapore(2002) 301–312
17. Wang H. Cao J. and Zhang Y.: Ticket-Based Service Access Scheme for Mobile Users. In Oudshoorn M. editor: Proceedings of Twenty-Fifth Australian Computer Science Conference. Melbourne, Victoria (2002) 178–187.