UML-Based Representation of Role-Based Access Control †

Michael E. Shin and Gail-Joon Ahn
ISE Department
George Mason University
4400 University Dr.
Fairfax, VA 22030, U.S.A.
{eshin,gahn}@isse.gmu.edu

Abstract

In role-based access control (RBAC) permissions are associated with roles, and users are made members of appropriate roles thereby acquiring the roles' permissions. The principal motivation behind RBAC is to simplify administration. Several frameworks for the development of role-based systems have been introduced. However, there are a few works specifying RBAC in a way which system developers or software engineers can easily understand and refer to develop role-based systems. The Unified Modeling Language (UML) is a general-purpose visual modeling language in which we can specify, visualize, and document the components of a software system. In this paper we represent the RBAC model with this well-known modeling language to reduce a gap between security models and system developments. We specify the RBAC model with three views: static view, functional view, and dynamic view. In addition, we briefly discuss about the future directions.

1. Introduction

In RBAC permissions are associated with roles, and users are made members of appropriate roles thereby acquiring the roles' permissions. This greatly simplifies management of permissions. Roles are created for the various job functions in an organization and users are assigned to roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. And the access of users to the information is regulated on the basis of the roles which are assigned to the users.

Since RBAC has become widely accepted as the proven technology, many security researchers and secure system developers have spent their time to develop role-based systems. Several frameworks for the development of role-based systems have been introduced [2, 11, 12]. These prior works were sometimes hard for system developers to understand because some are too abstract and formal, and others are ad-hoc solutions which are focused on application-oriented or domain-specific frameworks. These frameworks are not good enough to give a sound blueprint to system developers.

Our main objective here is to reduce such a gap between security models and system developments. In this paper we represent RBAC with a general-purpose visual modeling language UML. We choose the UML because it has been a standard language in the modeling community. Our representation includes static, functional, and dynamic views of RBAC model to achieve our objective.

This paper is organized as follows. In section 2 we describe a well-known model for role-based access control, commonly known as RBAC96. Section 3 briefly overviews UML. In section 4 we represent RBAC96 model with UML. Section 5 concludes this paper.

2. RBAC Model

RBAC has recently received considerable attention as a promising alternative to traditional discretionary and mandatory access controls (see, for example, [3, 4, 6, 9]). As MAC is used in the classical defense arena, the policy of access is based on the classification of objects such as top-secret level. The main idea of

†All correspondence should be addressed to: Dr. Gail-Joon Ahn, Information and Software Engineering Department, Mail Stop 4A4, George Mason University, Fairfax, VA 22030; email:gahn@isse.gmu.edu; phone:+1-703-993-1668; fax:+1-703-993-1638.

0-7695-0798-9/00 $10.00 © 2000 IEEE

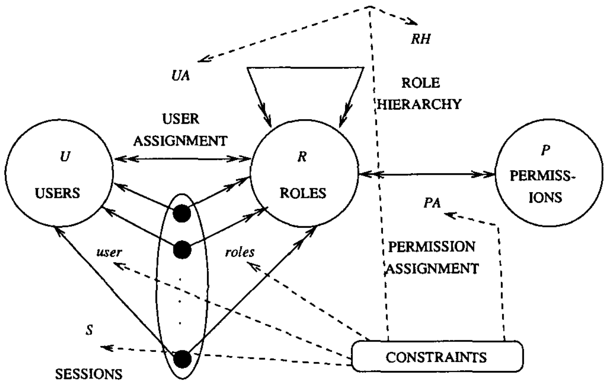

Figure 1: RBAC Model

DAC is that the owner of an object has discretionary authority over who else can access that object. But RBAC policy is based on the roles of the subjects and can specify security policy in a way that maps to an organization's structure.

A general family of RBAC models called RBAC96 was defined by Sandhu et al [9]. Figure 1 illustrates the most general model in this family. Motivation and discussion about various design decisions made in developing this family of models is given in [9].

Figure 1 shows (regular) roles and permissions that regulate access to data and resources. Intuitively, a user is a human being or an autonomous agent, a role is a job function or a job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role, and a permission is an approval of a particular mode of access to one or more objects in the system or some privilege to carry out specified actions. Roles are organized in a partial order $\geq$, so that if $x \geq y$ then role x inherits the permissions of role y. Members of x are also implicitly members of y. In such cases, we say x is senior to y. Each session relates one user to possibly many roles. The idea is that a user establishes a session and activates some subset of roles that he or she is a member of (directly or indirectly by means of the role hierarchy). The RBAC model has the following components and these components are formalized from the above discussions.

- U is a set of users,
- R is disjoint sets of roles and administrative roles respectively,
- P is disjoint sets of permissions and administrative permissions,
- $UA \subseteq U \times R$, is a many-to-many user to role assignment relation,
- $PA \subseteq P \times R$ is a many-to-many permission to role assignment relation,
- $RH \subseteq R \times R$ is partially ordered role hierarchies (written as $\geq$ in infix notation),
- S is a set of sessions,
- $user : S \rightarrow U$, is a function mapping each session s_i to the single user $user(s_i)$ and is constant for the session's lifetime,
- $roles : S \rightarrow 2^R$ is a function mapping each session s_i to a set of roles $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)[(user(s_i), r') \in UA]\}$ (which can change with time) so that session s_i has the permissions $\cup_{r \in roles(s_i)}\{p \mid (\exists r'' \leq r)[(p, r'') \in PA]\}$, and
- there is a collection of constraints stipulating which values of various components of the RBAC model are allowed or forbidden.

A user can be a member of many roles and a role can have many users. Similarly, a role can have many permissions and the same permissions can be assigned to many roles. Each session relates one user to possibly many roles. Intuitively, a user establishes a session during which the user activates some subset of roles that he or she is a member of. The permissions available to the users are the union of permissions from all roles activates in that session. Each session is associated with a single user. This association remains constant for the life of a session. A user may have multiple sessions open at the same time, each in a different window on the workstation screen for instance. Each session may have a different combination of active roles. The concept of a session equates to the traditional notation of a subject in access control. A subject is a unit of access control, and a user may have multiple subjects (or sessions) with different permissions active at the same time.

3. Overview of UML

The Unified Modeling Language (UML) is a general-purpose visual modeling language in which we can specify, visualize, and document the components

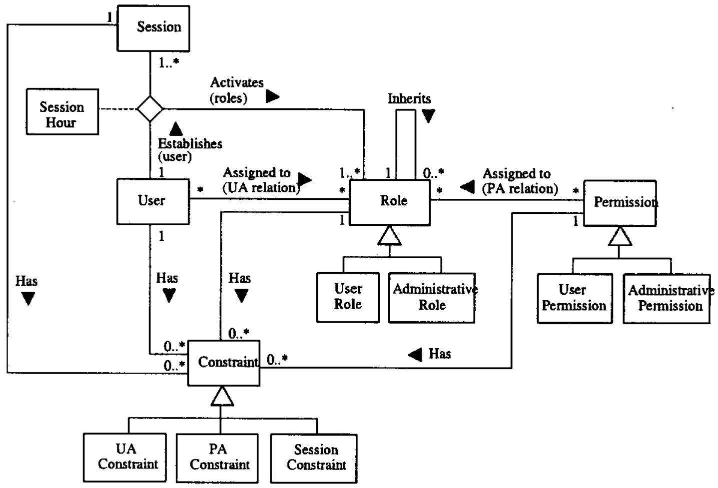

Figure 2: Class Diagram: Conceptual Static Model

of a software system. It captures decisions and understanding about systems that must be constructed [7]. It has been a standard language in the field of software engineering.

The UML consists of *use case modeling*, *static modeling*, and *dynamic modeling*. In *use case modeling*, the functional requirements of systems are specified with use cases and actors. A use case is initiated by actors and it defines interactions between the actors and the systems. *Static modeling* provides a structural view of information in the systems. In such a view, classes are defined in terms of attributes, as well as relationships with other classes. The relationships include association, generalization/specialization, and aggregation of classes. *Dynamic modeling* shows a behavioral view of the systems. It can be described with object collaboration diagrams, sequence diagrams, or statecharts. Object collaboration diagrams and sequence diagrams are developed to show how objects collaborate with each other to execute the use cases. State dependent views of objects are defined in statecharts [5].

In this paper, we take class diagrams, use case diagrams, and object collaboration diagrams for a static view, a functional view, and a dynamic view of the RBAC model, respectively. In the rest of this paper, we use UML notations which were introduced in [1, 7, 10].

4. UML-Based RBAC Presentation

Major components in RBAC are users, roles, permissions, sessions, and constraints. In order to represent RBAC model using UML, we analyze each component with a notion of object class. In the subsequent sections, our analysis is specified by three different views such as a static view, a functional view, and a dynamic view.

4.1. Static View

The conceptual static model for RBAC is depicted in Figure 2. It contains classes, relationships between classes, and cardinalities in relationships. The basic entities are user, role, permission, constraint, and session classes. The role and permission classes, respectively, may be specialized to two categories: user and administrative. This specialization depends on the level of users' qualification. The constraints in the RBAC model can have various forms, which are dependent on application systems. In order to simplify

«entity» User	«entity» Role	«entity» Permission	«entity» Session	«entity» SessionHour
user : String	role : String	permission : String	sessionName String	sessionName String
roleList List	subrole: String	roleList List	roleList List	startTime List

Figure 3: Attributes of Entity Classes

the analysis model, the constraint in our static model has only three constraints such as user constraint, permission constraint, and session constraint. Also, the static model has a special class called session hour. This class is used when a user establishes a session to activate her/his roles. This notion is useful to express session-based constraint. For example, an organization may require that a user can establish her/his session only during the certain amount of time. In order to enforce this kind of constraints, we need to keep track of session hours for each session. Attributes of entity classes are defined in Figure 3.

In the static model, UA relation and PA relation are represented as "Assinged to" relation with a many-to-many cardinality. User-session relation is viewed as a user can establish one or more sessions to activate at least one or more roles per each session with the constant session lifetime. The role inheritance relation is shown as a role inherits the other roles.

4.2. Functional View

In this paper, we also make more concrete functional requirements to represent the functions that RBAC systems should provide are not clearly defined in section 2. The functional view is depicted in Figure 4 using the use case model that has three actors such as a security administrator, a user, and a role domain engineer. The role domain engineer who extracts the foundational knowledge from application systems may organize a set of permissions, construct role hierarchies, and specify constraints. The security administrator who administrates a role-based system may assign users to roles and assign permissions to roles. The user who would be real persons or external systems may establish sessions, request permission approval, and close sessions.

The following shows the brief specification of the session establishing use case:

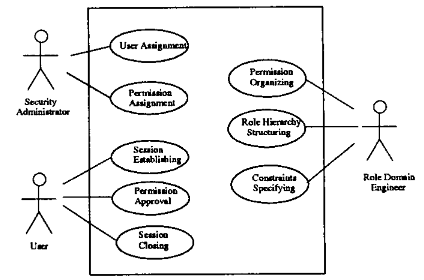

Figure 4: Use Case Model

> **Use case**: Session Establishing use case
> **Actors**: User
> **Precondition**: System idle
> **Description**: A user presents an information for establishing a session. System displays the roles that a user can activate. A user selects roles to activate. System activates a session with the roles that a user selected.

After a user establishes her/his session with selected roles, a user may need to access the system resources requiring authorization procedures that should be based on her/his role information. In other words, the permissions that are associated with her/his roles should be approved by the system. The following shows the brief specification of such a use case, called permission approval use case:

> **Use case**: Permission Approval use case
> **Actors**: User
> **Precondition**: A session was previously activated for a user.
> **Description**: A user presents an information for permission approval. System notifies a user whether or not the permission is approved.

In this paper, the limited functions are inferred from the RBAC model. We may also consider other situations because the functions in the RBAC system can be articulated. For example, we may require additional functions for monitoring sessions initiated by a security administrator or inquiring a user's status initiated by a user.

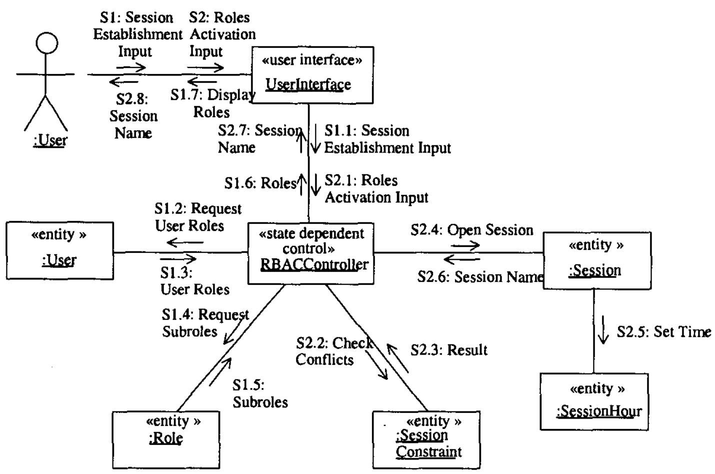

Figure 5: Collaboration Diagram: Session Establishment

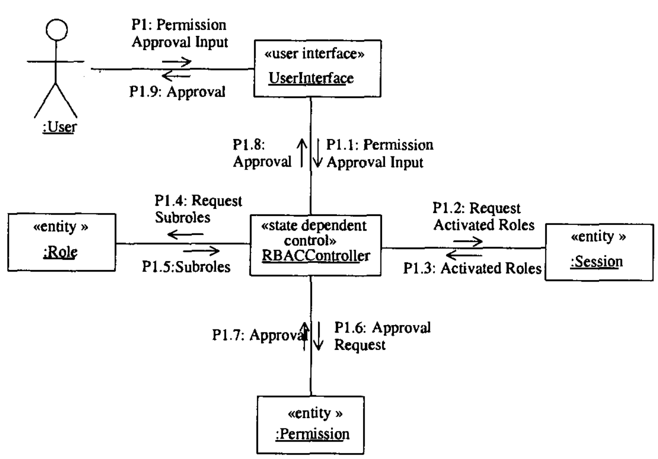

Figure 6: Collaboration Diagram: Permission Approval

4.3. Dynamic View

In the dynamic view, the use cases are refined to show the interactions among the objects that participate in each use case. The collaboration diagram for the session establishment is depicted in Figure 5 where a user initiates the use case through a user interface and RBACController coordinates interactions between the objects in the use case. The collaboration diagram for permission approval is illustrated in Figure 6 where it requires a precondition that a session has been activated before the execution of permission approval use case. The complete descriptions of each diagram are omitted for the simplicity.

5. Conclusion

In this paper, we briefly described a well-known model for role-based access control. We specified this model using the visual modeling language UML. Rather than simply enumerating each component in RBAC model, we showed UML-based analysis model using class diagrams, use case diagrams, and object collaboration diagrams. This is the first attempt for specializing RBAC model using a modeling language. We believe that our work can help system developers to understand RBAC model more easily and to build role-based systems. Also we could identify useful functions and constraints which were ruled out at the beginning of the security model.

Based on this work, we would investigate how the UML-based model can be accommodated to specify each component in RBAC model. It may include how to represent role hierarchies and constraints with some possible extensions of the UML. Because models help us understand the system by simplifying some of the details, this direction will be of practical interests.

Acknowledgements

The authors would like to thank Mr. Jaehong Park for his assistance on reviewing this paper.

References

[1] G. Booch, J. Rumbaugh, and I. Jacobson. The Unified Modeling Language User Guide. Addison Wesley, 1999.

[2] Pete Epstein and Ravi Sandhu. Towards A UML Based Approach to Role Engineering. In *Proceedings of the 4th ACM Workshop on Role-Based Access Control*, pages 135–142, Fairfax, VA, October 28-29 1999.

[3] David Ferraiolo and Richard Kuhn. Role-based access controls. In *Proceedings of 15th NIST-NCSC National Computer Security Conference*, pages 554–563, Baltimore, MD, October 13-16 1992.

[4] M.-Y. Hu, S.A. Demurjian, and T.C. Ting. User-role based security in the ADAM object-oriented design and analyses environment. In J. Biskup, M. Morgernstern, and C. Landwehr, editors, *Database Security VIII: Status and Prospects*. North-Holland, 1995.

[5] Hossan Gomaa. Object Oriented Analysis and Modeling for Family of systems with the UML. Technical Report, ISE Dept. George Mason University, 1999.

[6] Imtiaz Mohammed and David M. Dilts. Design for dynamic user-role-based security. *Computers & Security*, 13(8):661–671, 1994.

[7] J. Rumbaugh, G. Booch, and I. Jacobson. The Unified Modeling Language Reference Manual. Addison Wesley, Reading MA, 1999.

[8] Ravi S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[9] 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.

[10] National Software et al. Unified Modeling Language Notation Guide, Version 1.1. September 1, 1997.

[11] Charles Youman, Ed Coyne, and Ravi Sandhu, editors. *Proceedings of the 2nd ACM Workshop on Role-Based Access Control*, Nov. 6-7. ACM, 1997.

[12] Charles Youman, Ed Coyne, and Ravi Sandhu, editors. *Proceedings of the 3rd ACM Workshop on Role-Based Access Control*, Oct. 22-23. ACM, 1998.