8.2 Role-Based Access Control

This pattern describes how to assign rights based on the functions or tasks of people in an environment in which control of access to computing resources is required and where there is a large number of users, information types, or a large variety of resources. It describes how users can acquire rights based on their job functions or their assigned tasks.

Example

The hospital has many patients, doctors, nurses, and other personnel. The specific individuals also change frequently. Defining individual access rights has become a time-consuming activity, prone to errors.

Context

Any environment in which we need to control access to computing resources and where there is a large number of users, information types, or a large variety of resources.

Problem

For convenient administration of authorization rights we need to have ways to factor out rights. Otherwise, the number of individual rights is just too large, and granting rights to individual users would require storing many authorization rules, and it would be hard for administrators to keep track of these rules.

How do we assign rights based on the functions or tasks of people?

The solution to this problem must balance the following forces:

- In most organizations people can be classified according to their functions or tasks
- Common tasks require similar sets of rights
- We want to help the organization to define precise access rights for its members according to a need-to-know policy

Solution

Most organizations have a variety of job functions that require different skills and responsibilities. For security reasons, users should get rights based on their job

functions or their assigned tasks. This corresponds to the application of the need-to-know principle, a fundamental security policy [Sum97]. Job functions can be interpreted as roles that people play in performing their duties. In particular, Web-based systems have a variety of users: company employees, customers, partners, search engines, and so on.

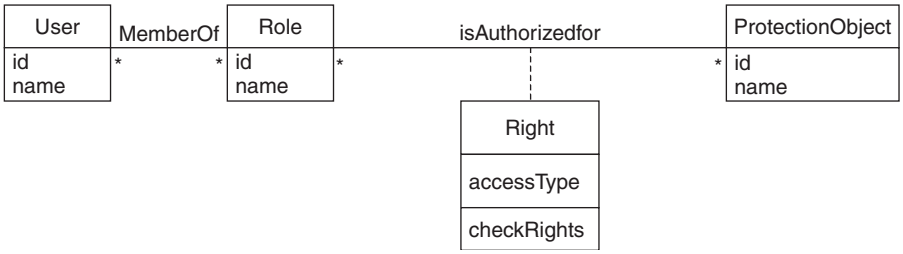
Structure

A class model for ROLE-BASED ACCESS CONTROL (249) (RBAC) is shown in the figure below. The `User` and `Role` classes describe the registered users and the predefined roles respectively. Users are assigned to roles, roles are given rights according to their functions. The association class `Right` defines the access types that a user within a role is authorized to apply to the protection object. In fact, the combination `Role`, `ProtectionObject`, and `Right` is an instance of AUTHORIZATION (245).

Implementation

Roles may correspond to job titles, for example manager, secretary. A finer approach is to make them correspond to tasks—for example, a professor has the roles of thesis advisor, teacher, committee member, researcher, and so on. An approach to define role rights is described in ROLE RIGHTS DEFINITION (259).

There are many possible ways to implement roles in a software system. [KBZ01]considers the implementation of the data structures needed to apply an RBAC model. Concrete implementations can be found in operating systems, database systems, and Web application servers.



Class model for ROLE-BASED ACCESS CONTROL (249)

Example Resolved

The hospital now assigns rights to the roles of doctors, nurses, and so on. The number of authorization rules has decreased dramatically as a result.

Variants

The model shown in the figure on page 252 additionally considers composite roles—it is an application of COMPOSITE [GoF95]—and separation of administration from other rights, an application of the policy of separation of duties. The administrator has the right to assign roles to groups and users, and is a special user who can assign users to roles and rights to a role. Rights for security administration usually include:

- Definition of authorization rules for roles
- Creation/deletion of user groups
- Assignment of users to roles

The figure also includes the concept of a Session, which corresponds to the way to use a role and can be used to enforce role exclusion at execution time. Finally, the `Group()` class describes groups of users that can be assigned to the same role.

Known Uses

Our pattern represents in object-oriented form a model described in set terms in [San96]. That model has been the basis of most research papers and implementations of this idea [FBK99]. RBAC is implemented in a variety of commercial systems, including Sun's J2EE [Jaw00], Microsoft's Windows 2000, IBM's WebSphere, and Oracle, amongst others. The basic security facilities of Java's JDK 1.2 have been shown to be able to support a rich variety of RBAC policies [Giu99].

Consequences

The following benefits may be expected from applying this pattern:

- It allows administrators to reduce the complexity of security, because there are much more users than roles.
- Organization policies about job functions can be reflected directly in the definition of roles and the assignment of users to roles.
- It is very simple to accommodate users arriving, leaving, or being reassigned. All these actions require only manipulation of the associations between users and roles.
- Roles can be structured for further flexibility and reduction of rules.
- Users can activate more than one session at a time for functional flexibility—some tasks may require multiple views or different types of actions.
- We can add UML constraints to indicate that some roles cannot be used in the same session or given to the same user (separation of duties).
- Groups of users can be used as role members, further reducing the number of authorization rules and the number of role assignments.

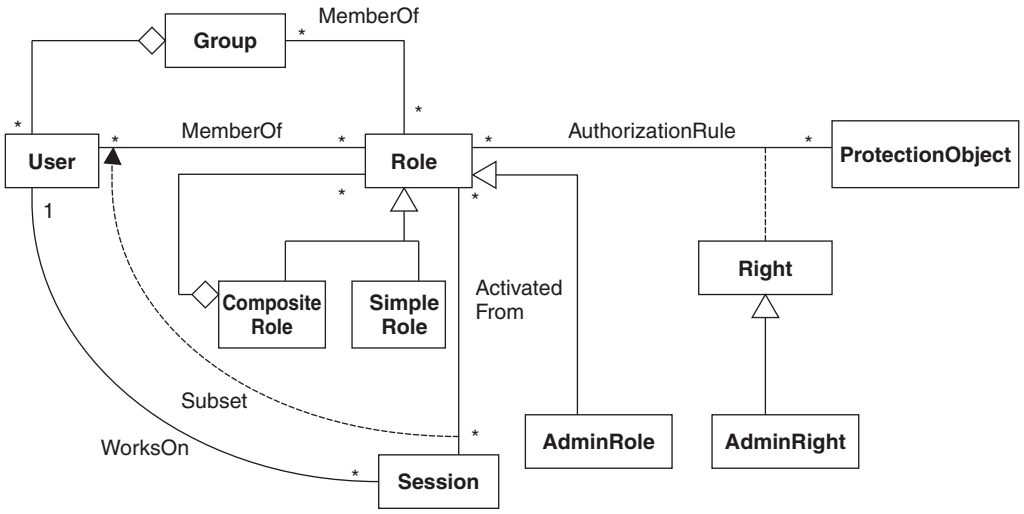
The following potential liability may arise from applying this pattern:

- Additional conceptual complexity—new concept of roles, assignments to multiple roles, and so on.

There are other possible structurings of roles [Fer94], which may be useful for specific environments. It is also possible to use roles to extend the multi-level model discussed in the next section.

See Also

Earlier versions of this pattern appeared in [Fer93] and [YB97], and a pattern language for its software implementation appears in [KBZ01], although this does not consider composite roles, groups, and sessions. The pattern shown in the figure below includes AUTHORIZATION (245) and COMPOSITE. Other related patterns are ROLE [Bau97], and ABSTRACT SESSION [Pry97].



A pattern for extended ROLE-BASED ACCESS CONTROL (249)