Chapter 10
Role-Based Access Control

Access is a specific type of interaction between a subject and an object that results in the flow of information from one to the other [177]. Following this definition, access control is the process of limiting access to the resources of a system only to authorised programs, processes, or other systems [177].

Role-based access control (RBAC) is a method to define and assess security access to different system artefacts, including data, functions and services. In this chapter we will introduce the major RBAC terms, and present the requirements for RBAC solution development and administration. Implementation of the RBAC solutions could be performed using model-driven security technologies. In the chapter we will discuss how SecureUML and UMLsec modelling languages could be applied when developing RBAC models and their applications.

10.1 Family of RBAC Models

The family of the role-based access control (RBAC) modelling consists of four models [65]: core RBAC, hierarchical RBAC, constrained RBAC and symmetric RBAC. In this section, firstly we will present the major RBAC concepts and principles (i.e., discuss the core RBAC model); then the extensions of the core RBAC are surveyed.

Core RBAC [65, 177] is illustrated in Fig. 10.1. A User is defined as a human being, but this concept could also be extended to machines, networks, or intelligent autonomous agents. A Role is a job function within the context of an organisation. Some associated semantics include authority and responsibility conferred on the user assigned to the role. Session is a mapping between a user and an activated subset of roles the user is assigned to. Permission is an approval to perform an operation on one or more protected objects. An Operation is an executable image of a program, which upon invocation executes some function for the user. Hence, the
operation types and secured objects depend on the type of the system where they are implemented. *User assignment* and *permission assignment* are many-to-many relationships. The first describes how users are assigned to their roles. The second characterises the set of privileges assigned to a role.

The core RBAC defines how users acquire permissions through roles, how user assignments and permission assignments are defined, monitored, and controlled. In this chapter will focus our discussion on the core RBAC model.

![Core RBAC](image)

**Fig. 10.1** Core RBAC, adapted from [65, 177].

**Extensions of core RBAC** are: (i) hierarchical RBAC, (ii) constrained RBAC, and (iii) symmetric RBAC [65, 177]. In addition to the core RBAC, the *hierarchical RBAC* includes role hierarchies. A *hierarchy* is a partial order defining a seniority relation between roles, whereby senior roles acquire the permissions of the juniors. For example, role $r_1$ inherits role $r_2$ only if all permissions of $r_2$ are also permissions of $r_1$ and all users of $r_1$ are also users of $r_2$. There are two types of hierarchy definition [65]: in *general hierarchical RBAC*, an arbitrary partial order is defined to support role hierarchy; in *restricted hierarchical RBAC*, some restrictions (e.g., tree or inverted tree structure) are imposed on the role hierarchy.

The *constrained RBAC*, in addition to the hierarchical RBAC, introduces constraints such as static separation of duty, dynamic separation of duty, prerequisite roles, and cardinality constraints [65]. *Separation of duties* spreads responsibility and authority for an action or task over multiple users, thereby raising the risk involved in committing a fraudulent act by requiring the involvement of more than one individual. The RBAC respects the static and dynamic separation of duties through user-role assignment and through role activation during the same session. The *prerequisite roles* are based on competency and the appropriateness needed. *Prerequisite constraints* require that a user be assigned to a role only if the user is already assigned to the role’s prerequisites. Finally, *cardinality constraints* are used to restrict the number of users assigned to a role, the number of roles a user can play, the number of roles a permission can be assigned to, or the number of sessions a user is allowed to activate at the same time.

Finally the *symmetric RBAC* extends the constrained RBAC by adding support for permission-role review with performance comparable to that of user-role review [65]. This means that the roles to which a particular permission is assigned can