5S−6

Role-Based Purpose-Oriented Access Control in Object-Oriented Environment[*]

Tsunetake Ishida, Hiroaki Higaki, and Makoto Takizawa[†]
Tokyo Denki University[‡]
Email : {tsune, hig, taki}@takilab.k.dendai.ac.jp

1 Introduction

Various kinds of object-oriented based systems, C++ and JAVA have been developed. Object-oriented systems are composed of multiple objects. An object is an encapsulation of data and methods for manipulating the data. The objects are structured with *is-a* relations in the object-oriented systems while the objects are not related with *is-a* in the object-based systems. The Common Object Request Broker Architecture (CORBA) is now getting a standard framework for realizing the interoperability among various kinds of distributed applications. In addition to realizing the interoperability, the applications are required to be secure, i.e. objects not only have to be protected from illegally manipulated but also have to be prevented illegal information flow among objects. In section 2, we present the role concept the object-oriented system. In section 3, we discuss the purpose-oriented access control model. In section 4, we discuss information flow.

2 System Model

2.1 Object-oriented system

Object-oriented systems are composed of objects. Objects are encapsulations of data and methods for manipulating the data. There are two kinds of objects; *classes* and *instances*. A class is defined to be a set of *attributes* and *methods*. An instance is a tuple of values, each of which is a value of an attribute in the class, with the methods of the class. A term "object" means an instance in most object-oriented systems like JAVA and C++.

A method of an object is invoked by sending a request message to the object. The method specified in the request message is performed on the object on receipt of the request. Then, the object sends the response back to the sender of the message. The method may further invoke methods in other objects. Thus, the invocations of the methods are *nested*.

A class can be derived from one or more classes. Here, suppose a class c_2 is derived from a class c_1. c_2 is referred to as a *subclass* of c_1. In turn, c_1 is a *supperclass* of c_2. The class c_2 inherits the attributes and methods of c_1. Here, c_2 *is-a* c_1. *Inheritance* provides means for building new classes from the existing classes. A class may *override* the definition of attributes and methods inherited from the supperclass.

2.2 Roles

Each subject plays some *role* in an organization, e.g. professor, assistant, and student in a university. A role represents a job function that describes the authority and responsibility in the organization. Each person is assigned some role and then plays the role in the organization. In the *role-based access control* model [1], a *role* is modeled to be a set of *access rights*. An access right is given a pair of a method *op* and an object *o* which supports *op*, i.e. $\langle o, op\rangle$. That is, a role means what method can be performed on what object. A subject s is granted a role r only if s plays the role r in the organization. On the other hand, each access right is granted to subjects in the access control model. Here, a subject s is referred to as *bound* with the role r if r is granted to s. This means that s can perform a method *op* on an object o if $\langle o, op\rangle \in r$. If a subject s would like to exercise the authority of a role r with which s is bound, the subject s first establishes a *session* to the role r. Then, s can play a role of r, i.e. s can manipulate o by *op*. The number of sessions which each subject can establish at the same time can be restricted to be one.

Some roles are *hierarchically* structured to represent logical authority and responsibility in an organization. If a role r_i includes every access right of another role r_j, r_i is referred to as *higher than* r_j ($r_j \preceq r_i$). The relation "$\preceq$" is transitive. Here, the roles r_i and r_j are *uncomparable* if neither $r_j \preceq r_i$ nor $r_i \preceq r_j$.

3 Purpose-Oriented Access Control

3.1 Purpose concept

The purpose-oriented model [2] newly introduces a *purpose* concept to the access control model. A *purpose* shows why each subject s manipulates an object o by invoking a method *op* of o. In the object-based system, methods are invoked in the nested manner. Suppose that a subject s invokes a method op_1 of an object o_1 and then op_1 invokes a method op_2 of an object o_2. In the purpose-oriented model, the method op_1 invoking a method op_2 of an object o_2 shows purpose for what the object o_1 manipulates the other object o_2, while the access control model specifies whether or not o_1 can manipulate o_2 by invoking op_2. Finally, the method op_1 of the object o_1 can invoke op_2 of o_2 only if the access rule $\langle o_1 : op_1,\ o_2 : op_2\rangle$ is specified.

3.2 Hierarchical system

A role is specified in a collection of access rights in the role-based model [1]. We extend the purpose-oriented access control model to incorporate the role concept. In the object-based system, objects are related in the invocation relation. In this paper, the objects can be classified into some levels. Objects at a level can invoke methods supported by objects of a lower level but cannot invoke objects of a higher level. Such a system is referred to as *hierarchical* system.

An object o_1 is *higher* than another object o_2 ($o_1 \succ o_2$) iff a method of o_1 invokes a method of o_2 or $o_1 \succ o_3 \succ o_2$ for some object o_3. The objects are hierarchically structured in the system iff $o \succ o$ does

[*] オブジェクト指向環境における役割に基づいた目的指向アクセス制御

[†] 石田 常竹, 桧垣 博章, 滝沢 誠

[‡] 東京電機大学

not hold for every object o, i.e. $\succ$ is irreflexive. A pair of objects o_1 and o_2 are at the same level ($o_1 \equiv o_2$) iff neither $o_1 \succ o_2$ nor $o_2 \succ o_1$. Objects at the level 0 are objects which are not invoked by any other objects. The *Consumer* objects are at level 0. Objects at the level i are objects which are invoked by objects at the level $i-1$. In this paper, we assume that each object belongs to one level. That is, each object at a level i invokes only methods of objects at the level $i+1$. Objects which do not invoke methods of other objects are at the lowest level and named *primitive* objects. A *hierarchical* system is one where the objects are hierarchically structured. In this paper, we consider a system where objects are *hierarchically* structured in the invocation relation.

We consider roles in a hierarchical system. A role of a level i is a collection of access rights on the objects at the level i. Let R^i be a role of a level i which is $\{ \langle o^i, op \rangle \mid o^i$ is an object of the level i and op is a method of o_i $\}$.

Suppose that a method op_1 of an object o_1 invokes op_2 of o_2. Here, o_1 is at a level i and o_2 is at level $i+1$. We also suppose that an access right $\langle o_1, op_1 \rangle$ is in a role R_1. The method op_1 invokes a method op_2 of an object o_2 which is at level $i+1$. At level $i+1$, the access right $\langle o_2, op_2 \rangle$ is included in roles R_2^{i+1} and R_3^{i+1}. If an object o_1 is bounded with a role R_2^{i+1}, o_1 is allowed to invoke the method op_2 of the object o_2. This is a simple extension of the role-based model to the hierarchical object-based system.

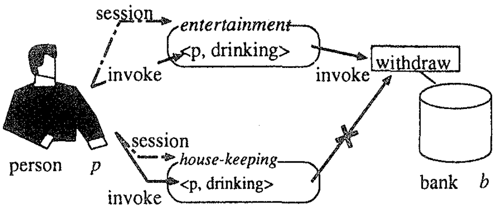
Each method op_i of an object o_i is granted a role $r_i = \{\langle o_{i1}, op_{i1} \rangle, \ldots, \langle o_{ih_i}, op_{ih_i} \rangle\}$. This means, op_i can invoke a method op_{ij} of an object o_{ij} (for $j = 1, \ldots, h_i$). In turn, op_{ij} may be granted a role $r_{ij} = \{\langle o_{ij1}, op_{ij1} \rangle, \ldots, \langle o_{ijh_{ij}}, op_{ijh_{ij}} \rangle\}$. op_{ij} can invoke op_{ijk} of o_{ijk} if op_{ij} is granted r_{ij}. An access rule has to show in what role the method op_i of the object o_i is bound with the role r_i.

[Purpose-oriented role-based access (POR) rule] $\langle r : o_i : op_i, r_i \rangle$ means that a method op_i of an object o_i is invoked in a role r and op_i can invoke methods specified in a role r_i. □

[Example] Suppose that there are two roles *entertainment* and *house-keeping* including access right $\langle p$, *drinking*$\rangle$ and $\langle p$, *shopping*$\rangle$, respectively. A person p plays the roles in a community and manipulates the bank object b by authority of its role. If the method *drinking* of p is invoked in the role *entertainment*, p is allowed to withdraw money from the bank b. However, p is not allowed to do so if *drinking* of p is invoked in the role *house-keeping*. Thus, the access rule is specified in a form $\langle$*entertainment* : p : *drinking*, b : *withdraw*$\rangle$ where the method *drinking* shows the *purpose* of p. □

3.3 Class role

The object-oriented system is composed of classes and objects, i.e. instances of the classes. There are two kinds of access rights, *class* and *instance* access rights. A class access right is in a form $\langle c, op \rangle$ where c is a class and op is a method of the class c. On the other hand, an instance access right is in a form $\langle o, op \rangle$ where o is an object and op is the method of o. There are two kinds of roles, i.e. class roles and instance roles. A class role r is defined in terms of methods and classes, i.e. $r = \{\langle c, op \rangle\}$. On the other hand, an instance role r' is defined in terms of methods and objects, i.e. $r' = \{\langle o, op \rangle\}$. r' is instantiated from the class role r. In the instance role r', o is an object which is instantiated from a class c.



Purpose-oriented role-based access rule :
<entertainment : p : drinking, b : withdraw>

Figure 1: Purpose-oriented role-based access.

Furthermore, there is an *is-a* relation in object-oriented systems. The *is-a* relation is defined among classes. We extend the role concept to conform to the *is-a* relation. Suppose that there are two classes c_1 and c_2. The class c_2 is defined as a specialization of the class c_1, i.e. c_2 *is a* c_1. The access right $\langle c_2$, $op\rangle$ is automatically included in the role r where r is given as $\{\langle c_1, op \rangle\}$. This means that the access right of specialized class is given to the role when the role has an access right of its supperclass.

4 Information Flow Control

In the role-based access control model presented in the previous section, it is assured that subjects manipulate objects based on roles to which the subjects belong. However, illegal information flow among objects may occur. Because legal and illegal information flow among the objects are not discussed. For example, suppose that a subject s_i invokes *write* on an object o_j after invoking *read* on o_i by the authority of a role r_i. This means that s_i may write data obtained from o_i to o_j. s_j can read data in o_i even if read access right is not authorize to a role r_j. This is the confinement problem pointed out in the basic access control model. In addition, a subject can have multiple roles in the role-based model even if they can play only one role at the same time. We classify methods of objects with respect to the following points: 1. whether or not a value v_i of attribute a_i from an object o_i is output. 2. whether or not a value of a_i in o_i with input parameter is changed.

The methods are classified into four types in 1) m_R, 2) m_W, 3) m_{RW}, and 4) m_N. m_R means that the method outputs value but does not change o_i. m_W means that the method does not output but changes o_i. The method m_{RW} outputs value and changes o_i. The method m_N neither output value nor changes o_i.

5 Concluding Remarks

This paper has presented an access control model for distributed object-oriented systems with role concepts. Roles are higher level representation of access control models. We have defined a role to mean what method can be performed on which object. Furthermore, we have discussed how to control information flow to occur through roles.

References

[1] Ferraiolo, D. and Kuhn, R., "Role-Based Access Controls," *Proc. of 15th NIST-NCSC Nat'l Computer Security Conf.*, 1992, pp. 554–563.

[2] Yasuda, M., Higaki, H., and Takizawa, M., "A Purpose-Oriented Access Control Model for Information Flow Management," *Proceeding of 14th IFIP Int'l Information Security Conf. (SEC'98)*, 1998, pp. 230–239.