Role-Based Access Control for Distributed Systems *

6G−1

Masashi Yasuda, Hiroaki Higaki, and Makoto Takizawa †
Tokyo Denki University ‡
Email {masa, hig, taki}@takilab.k.dendai.ac.jp

1 Introduction

By using object-oriented technologies, lots of object-oriented systems like object-oriented database management systems and languages like JAVA have been developed. Object-oriented systems are composed of multiple objects which cooperate to achieve some objectives by message passing. 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 system has to be secure. In the secure system, it is required to not only protect objects from illegally accessed but also prevent illegal information flow [2] among objects in the system. In this paper, we discuss a high assurance access control model for object-oriented systems.

In this paper, we discuss role concepts in the object-oriented model. Then, we discuss information flow to occur among the roles through the nested invocations.

2 System Model

2.1 Object-oriented system

Object-oriented systems are composed of objects. Objects are encapsulations of data and procedures for manipulating the data. Each object is associated with a unique identifier in the system. For each object, a set of *attributes* that specify the object structure, a set of *values* that specify the object state, and a set of *methods* that specify the object behavior are defined. An object o is defined as follows : (1) unique object identifier (*OID*), (2) set of attributes $(a_1, \ldots, a_n)$, (3) set of values $(v_1, \ldots, v_n)$ where each v_i is a value of a_i, and (4) set of methods $(t_1, \ldots, t_n)$. A *class* is an abstraction mechanism, which defines a set of similar objects sharing the same structure and behavior. Each object in the system is an *instance* of some class. A class shows a template for its instances. A method of an object is invoked by sending a message to the object. On receipt of the message, the object starts to compute the method specified by the message. On completion of the computation of the method, the object sends the response back to the sender object of the message.

In the object-oriented system, a *subject* shows a user or an application program. A subject is an active entity in the system. A subject manipulates an object by invoking its method to achieve some objectives. On the other hand, an *object* is a passive entity. An object activates a method only if the method is invoked on receipt of the message. A method invoked may invoke furthermore methods of other objects. Thus, the invocation is nested.

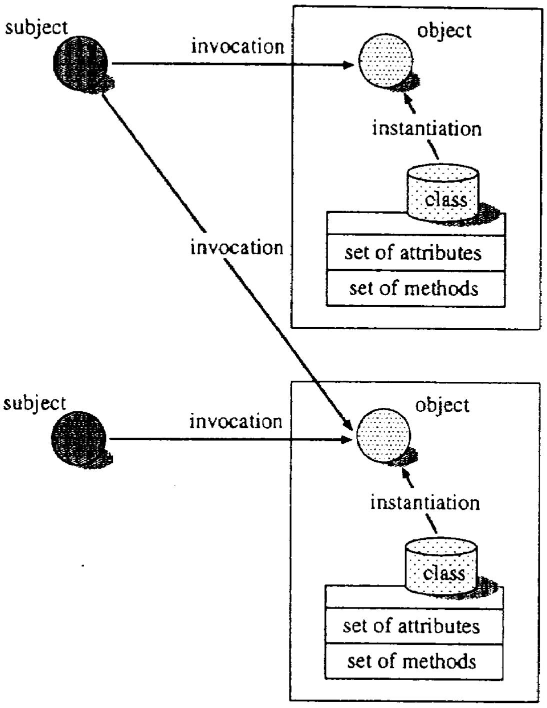

Figure 1: System model.

2.2 Roles

Each subject plays a *role* in an organization, like a designer and clerk. A role represents a job function that describes the authority and responsibility in the organization. In the role-based model [1, 4], a role is specified in a set of *permissions*. A permission means an approval of a particular mode of access, i.e. methods to an object in the system. That is, a role means what method can be executed on which object.

[Definition] A role $r \in R$ is a collection $\{\langle o, p\rangle\} \subseteq O \times P$. Here, R, O, and P show sets of roles, objects, and permissions in the system, respectively. □

A subject s is bound with a role r. Here, s is referred to *as belong* to r. This means that s can perform a method p on an object o if $\langle o, p\rangle \in r$.

Some roles are *hierarchically* structured to show structural authorizations in the system. A role hierarchy represents organization's logical authority and responsibility. If a role r_i is higher than r_j ($r_j \preceq r_i$), $r_j \subseteq r_i$. That is, r_i has all of permissions of lower role r_j, and i.e. more permissions than r_j.

3 Access Control

In a role-based model, subjects access to objects through roles that subjects belong to. A subject manipulates an object by invoking its method. An object activates the method only if the method is invoked by a subject. If a subject would like to exercise the authority of roles which they belong to, the subject establishes *sessions* to its roles.

[Definition] A subject s can access to an object o by

*分散システムのための役割に基づいたアクセス制御
†安田 昌史, 桧垣 博章, 滝沢 誠
‡東京電機大学

invoking a method p iff

(1) an owner of o assignes a permission p to a role r,

(2) s belongs to a role r, and

(3) s is establishing a session to r. □

For example, in Figure 2, a subject s can perform *write* on an object o while a session between s and a role *chief* is established. Even if s belongs to both roles *chief* and *clerk*, s cannot execute *write* on o if a session between s and *chief* is not established. The authority of a role r can be exercised only while a subject s establishes a session to r.

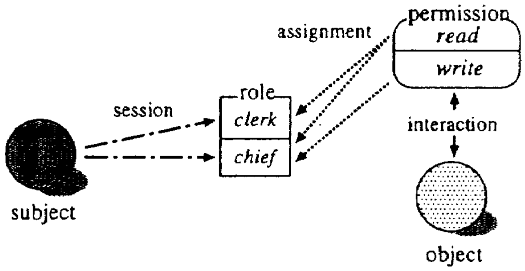

Figure 2: Role-based access.

4 Information Flow Control

In the role-based access control presented in the previous section, it is assured that subjects access to objects based on roles to which the subjects belong. However, illegal information flow among objects may occur. Because legal and illegal information flow are not defined. For example, in Figure 3, 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 permission 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. In Firue ??, suppose that a person A belongs to two roles *chief* and *clerk*. A obtains some information from *book* as a *clerk* and then stores the data derived from the information into book as a *chief*.

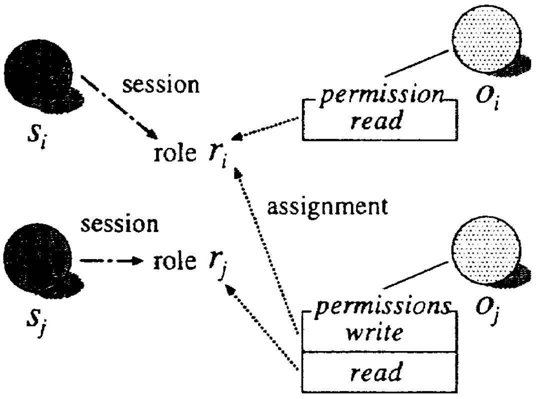

Figure 3: Illegal information flow.

We classify methods of objects with respect to the following points:

(1) whether or not outputs value v_i of attribute a_i from an object o_i.

(2) whether or not changes a value of a_i in o_i with input parameter.

The methods are classified into four types in (1) m_R, (2) m_W, (3) m_{RW}, and (4) m_N. m_R means the method output a value but does not change o_i. m_W means that the method does not output but change o_i. m_{RW} method outputs a value and changes o_i. m_N method neither outputs a value nor change o_i. For example, a *count-up* method is classified to be m_N because *count-up* change the state of object but does not need input parameter. *count-up* does not flow information into an object.

[Example 1] Let us consider a simple example about information flow between two objects o_i and o_j in Figure 4. A subject s is now in a session with a role r_i. Here, s can invoke method classified into m_R on o_i and m_{RW} on o_j by the authority of r_i, respectively. If s obtains information from o_i through m_R, s can invoke m_{RW} on o_j after the invocation of m_R on o_i. Because a set of roles on o_i which is authorized to execute methods classified into m_R is a subset of roles on o_j which is authorized to execute methods classified into m_R. □

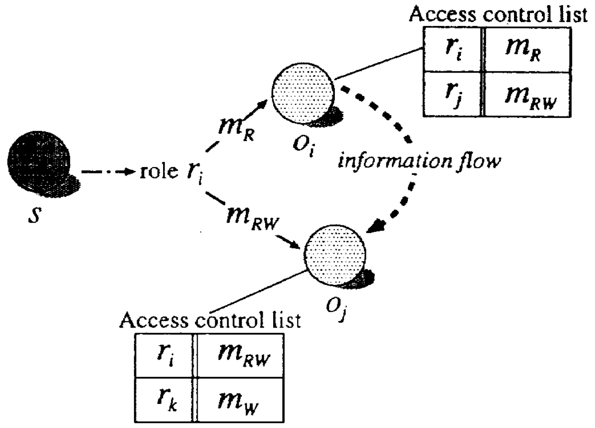

Figure 4: Information flow control.

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 executed 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] Lampson, B. W., "A Note on the Confinement Problem," *Communication of the ACM*, Vol. 16, No. 10, 1973, pp. 613-615.

[3] Sandhu, R. S., "Lattice-Based Access Control Models," *IEEE Computer*, Vol. 26, No. 11, 1993, pp. 9-19.

[4] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E., "Role-Based Access Control Models," *IEEE Computer*, Vol. 29, No. 2, 1996, pp. 38-47.

[5] Tachikawa, T., Yasuda, M., and Takizawa, M., "A Purpose-oriented Access Control Model in Object-based Systems," *Trans. of IPSJ*, Vol. 38, No. 11, 1997, pp. 2362-2369.