Role-Based Access Control for Distributed Objects

Masashi Yasuda, Tsunetake Ishida, Hiroaki Higaki, and Makoto Takizawa
Dept. of Computers and Systems Engineering
Tokyo Denki University
Email {masa, tsune, hig, taki}@takilab.k.dendai.ac.jp

Abstract

Various kinds of distributed applications have been developed by using object-oriented technologies. Object-oriented technologies are used to realize the interoperability of the applications. Object-oriented systems are composed of multiple objects which cooperate to achieve some objectives by passing messages. In addition to realizing the interoperability, it is essential to make the system secure. The secure system is required to not only protect objects from illegally manipulated but also prevent illegal information flow among objects. In this paper, we discuss role-based access control model in the object-oriented systems and how to resolve illegal information flow.

1. Introduction

By using object-oriented technologies, various kinds of object-oriented systems like object-oriented database management systems [2] and languages like JAVA [9] have been developed. Object-oriented systems are composed of multiple objects which cooperate to achieve some objectives by passing messages. An object is an encapsulation of data and methods for manipulating the data. The Common Object Request Broker Architecture (CORBA) [12] is now getting a standard framework for realizing the interoperability among various kinds of distributed applications. In addition to realizing the interoperability, secure system is required to not only protect objects from illegally manipulated but also prevent illegal information flow [4, 6, 13] among objects in the system.

In the basic access control model [10], an access rule is specified in a form $\langle s, o, t \rangle$ which means that a subject s can manipulate an object o in a type t of access. A pair $\langle o, t \rangle$ is an *access right* granted to s. Only the access request which satisfies the access rules specified by the authorizer is accepted to be computed. However, the access control model implies the *confinement* problem [11], i.e. illegal information flow may occur among subjects and objects. In order to make every information flow legal in the system, the *mandatory* access control model [1, 4, 13] is proposed. The legal information flow is given by classifying objects and subjects and defining the *can-flow* relation [4] between classes of objects and subjects. In the mandatory model, the access rules are specified by the authorizer so that only the legal information flow occurs. For example, if a subject s reads an object o, information in o flows to s. Hence, s can read o only if a *can-flow* relation from o to s is specified. In the discretionary model [3, 5, 6], the access rules are defined in a distributed manner while the mandatory access rules are specified only by the authorizer in a centralized manner. For example, the access rules can be granted to other subjects in the relational database Sybase [15]. In the role-based model [7, 14, 17], a *role* is defined to be a collection of access rights, i.e. pairs of access types and objects which show a job function in the enterprise. The access rule is specified by granting subjects the roles while each subject is granted an access right in the access control model.

The traditional models discuss what object can be manipulated by what subject in what access type. The authors [16, 18] newly propose a *purpose-oriented* model which takes into account a *purpose* concept why each subject manipulates objects in the object-based system. The purpose is modeled to be a method which invokes another method in the object-based system. In the object-based system, methods are invoked in a nested manner. It is critical to discuss how to specify access rules in the nested invocation of methods. One way is that a method op_1 of an object o_1 can invoke a method op_2 of an object o_2 if a subject which invokes op_1 is granted an access right $\langle o_2, op_2 \rangle$. Sybase [15] adopts the *ownership chain* mechanism where op_1 can invoke op_2 if the owner of o_2 is the same as o_1 even if s is not granted an access right $\langle o_2, op_2 \rangle$. It is not easy, possibly impossible to specify access rules for huge number of objects and subjects. Another way is that op_1 can invoke op_2 only if o_1 has an access right $\langle o_2, op_2 \rangle$. We take this approach, i.e. *object pairwise approach*. In addition, we discuss how to incorporate the role concepts into the purpose-oriented model in an object-oriented system where methods are invoked in

the nested manner. Then, we discuss information flow to occur among the roles through the nested invocations.

In section 2, we present the model in the object-oriented systems. In section 3, we discuss access rules. 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. 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, ..., a_n), 3) set of values (v_1, ..., v_n) where each v_i is a value of a_i, and 4) set of methods (t_1, ..., t_n). A *class* is an abstraction mechanism, which defines a set of similar objects sharing the same structure and behavior, which is given a set of attributes and methods. Each object in the system is an *instance* of some class [Figure 1]. A method of an object is invoked by sending a request 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.

We define *reliable* objects as follows :

[Definition] An object o is *reliable* if and only if (*iff*) the following conditions are specified :

1. o can be manipulated only through methods supported by o, and
2. no methods malfunction. □

We assume that every object is *reliable* in the system.

A class can be defined as a specialization of one or more classes. *Inheritance* provides means for building new classes from the existing classes. A class c defined as a specialization of a class c' is called a *subclass* of c' and inherits attributes and methods from c'. In turn, c is referred to as a *supperclass* of c'. An *is-a* relation is defined between a pair of superclass and subclass. A subclass may *override* the definition of attributes and methods from the supperclass. In Figure 2, classes *Clock* and *Alarm* are superclasses of a class *AlarmClock*. *AlarmClock* inherits attributes *time* and *setAlarm* from *Clock* and *Alarm*, respectively. *AlarmClock* also inherits methods *show* from *Clock* and the other methods *set* and *ring* from *Alarm*.

In the object-oriented system, a *subject* shows a user or an application program. A subject is an active entity in the system, which can issue access request to objects. The

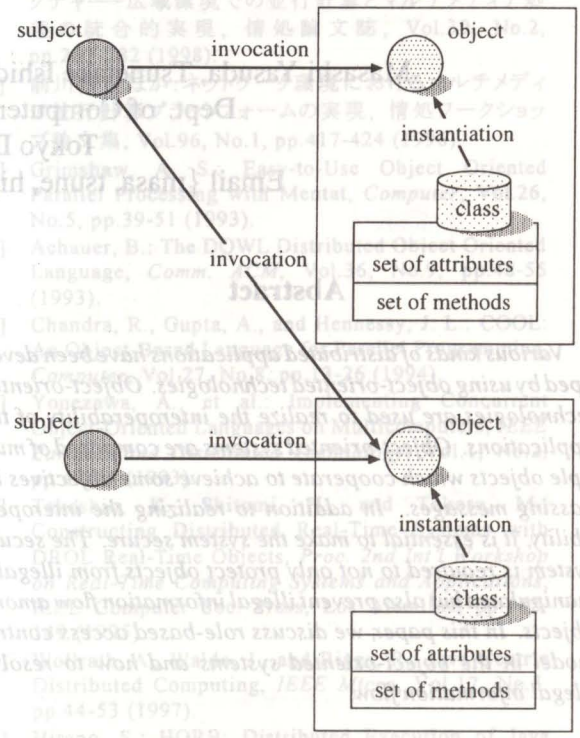


Figure 1. System model.

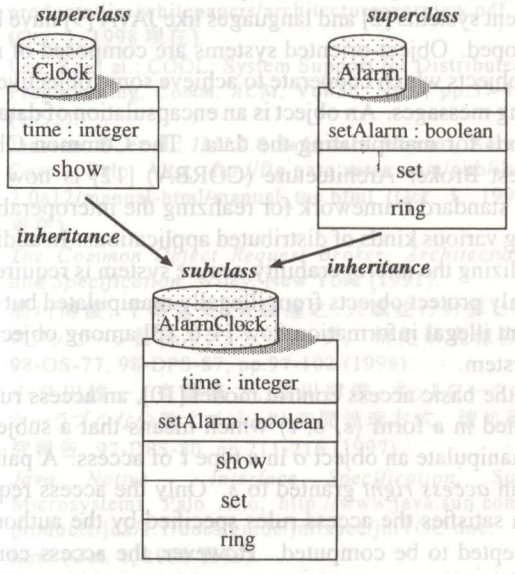


Figure 2. Class hierarchy.

subject manipulates objects by invoking their methods. 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. The method invoked may invoke further methods of other objects. Thus, the invocation is *nested.*

2.2. Roles

Each subject plays some *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 [7, 14, 17], a *role* is modeled in a set of *access rights.* An access right 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 performed on which object.

[Definition] A role r is a collection of access rights $\{\langle o, op\rangle\} \subseteq O \times M$ where O and M show sets of objects and methods in the system, respectively. □

Let R be a set of roles in the system. A pair $\langle o, op\rangle$ of an object o and a method op of o is a access right. In the role-based model, a subject s is granted roles while s is granted access rights in the access control model. Here a subject s is referred to as *bound* with the role r. Here, s is referred to *as belong* to r. This means that s can perform a method op on an object o if $\langle o, op\rangle \in r$. For example, a role *chief* is $\{\langle book, read\rangle, \langle book, enter\rangle\}$ and *clerk* is $\{\langle book, read\rangle\}$ in Figure 3. A person A who works as a chief in the company is granted the role *chief* in the organization. A *clerk* B is granted a role *clerk.* Thus, it is easy to grant access rights to persons.

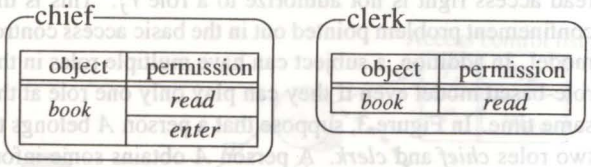
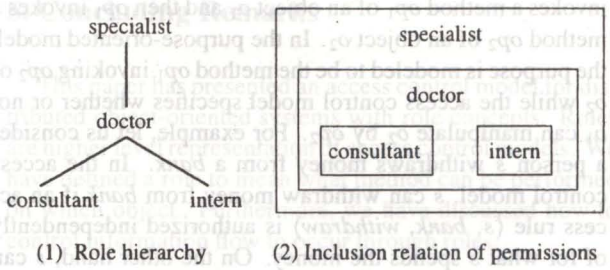


Figure 3. Roles.

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 includes all of access rights of another role r_j, r_i is *higher than* r_j ($r_j \preceq r_i$). $\preceq$ is transitive. In Figure 3, *clerk* $\preceq$ *chief* since *chief* takes a higher position than *clerk.* Figure 4 shows an example of the role-hierarchy. Here, *specialist* $\succeq$ *doctor*, *doctor* $\succeq$ *consultant*, and *doctor* $\succeq$ *intern.* *consultant* and *intern* are not related on $\preceq$.



(1) Role hierarchy (2) Inclusion relation of permissions

Figure 4. Role hierarchy and inclusion relation.

3. Access Control

In a role-based model, each subject s can manipulate an object o by a method op of o only if s is granted a role including an access right $\langle o, op\rangle$. The object activates the method on receipt of the request message. If a subject s would like to exercise the authority of a role r which s belongs to, the subject s establishes *sessions* to the role r.

[Access condition] A subject s can manipulate an object o by invoking a method op of o if

1. the owner of o assigns an access right op to a role r,
2. s belongs to a role r, and
3. s is establishing a session to r. □

For example, in Figure 5, 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 perform *write* on the object 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.

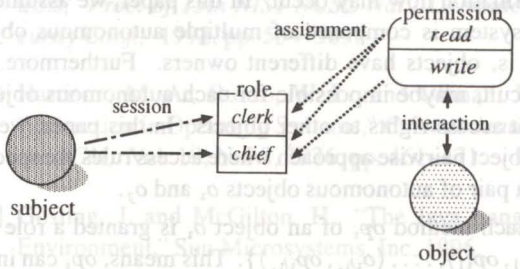


Figure 5. Role-based access.

The purpose-oriented model [16, 18] 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 purpose is modeled to be the method op_1 invoking op_2 of o_2 while the access control model specifies whether or not o_1 can manipulate o_2 by op_2. For example, let us consider a person s withdraws money from a *bank*. In the access control model, s can withdraw money from *bank* if an access rule $\langle s,$ *bank*, *withdraw*$\rangle$ is authorized independently of for what s spends the money. On the other hand, s can get money from *bank* for purpose of *house-keeping* but not for *drinking*. An access rule $\langle s$: *house-keeping*, *bank* : *withdraw*$\rangle$ is specified where a method *house-keeping* of s shows the purpose.

A role is specified in a collection of access rights in the role-based model [7, 14, 17]. We would like to extend the purpose-oriented access control model to the role-based model. In the object-based system, methods are invoked in a nested manner. Here, suppose that a subject s invokes a method op_1 on an object o_1 and then op_1 invokes another method op_2 on an object o_2. Here, suppose s is granted an access right $\langle o_1, op_1\rangle$. In one way, only if s is granted an access right $\langle o_2, op_2\rangle$, op_1 can invoke op_2. However, it is cumbersome for each object o_i to specify which subject can manipulate o_i. In the relational database management system Sybase [15], the ownership chain method is adopted. Here, if o_2 has the same owner as o_1 and s is granted an access right $\langle o_1, op_1\rangle$, op_1 can invoke op_2 even if s is not granted an access right $\langle o_2, op_2\rangle$. Otherwise, op_1 is allowed to invoke op_2 only if s is granted an access right $\langle o_2, op_2\rangle$. Suppose the response of op_2 carries some data stored in the object o_2. On receipt of the response, the object o_2 may store the data carried by the response in the storage while o_2 continues to compute op_1 by using the response. This means, information in o_2 flows to o_1 through the invocation. The data may be brought to other objects by further invocation. By using the ownership chain method, illegal information flow may occur. In this paper, we assume that the system is composed of multiple autonomous objects, that is, objects have different owners. Furthermore, it is difficult, maybe impossible for each autonomous object to grant access rights to other objects. In this paper, we take an object pairwise approach where access rules are specified for a pair of autonomous objects o_i and o_j.

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 a method op_{ijk} of o_{ijk} if op_{ij} is granted the role r_{ij}. An access rule has to show in what role the method op_i of the object o_i is bound to 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.

The object-oriented system is composed of classes and objects, i.e. instances of the classes. 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. $r = \{\langle c, op\rangle$ where c is a class and op is a method of $c\}$. On the other hand, an instance role r' is defined in terms of methods and objects, i.e. $r' = \{\langle o, op\rangle$ where c is an object and op is a method of $o\}$. r' is instantiated from the class role r.

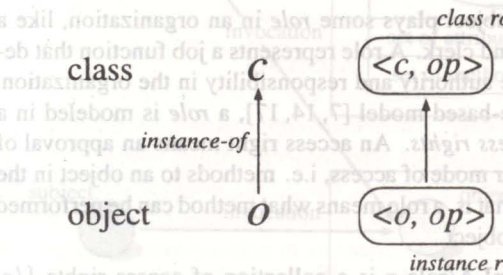


Figure 6. class role and instance role.

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, in Figure 7, 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. In Figure 3, suppose that a person A belongs to two roles *chief* and *clerk*. A person A obtains some information from *book* as a *clerk* and then stores the data derived from the information into book as a *chief*.

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 a 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

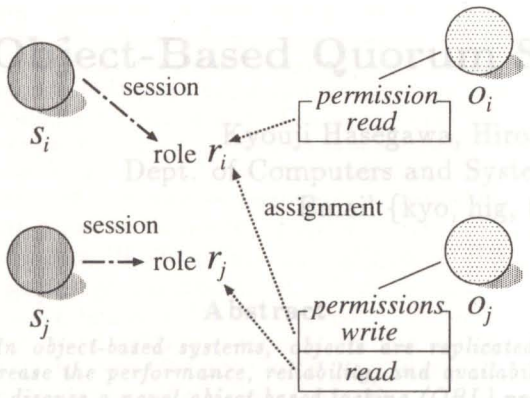


Figure 7. Illegal information flow.

a value and changes o_i. The method m_N neither outputs a value nor changes o_i. For example, a method *count-up* is classified to be m_N because *count-up* changes the state of the object but does not need input parameter. *count-up* does not bring information into an object.

[Example 1] Let us consider a simple example about information flow between a pair of objects o_i and o_j in shown Figure 8. A subject s is now in a session with a role r_i. Here, s can invoke methods 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 perform methods classified into m_R. □

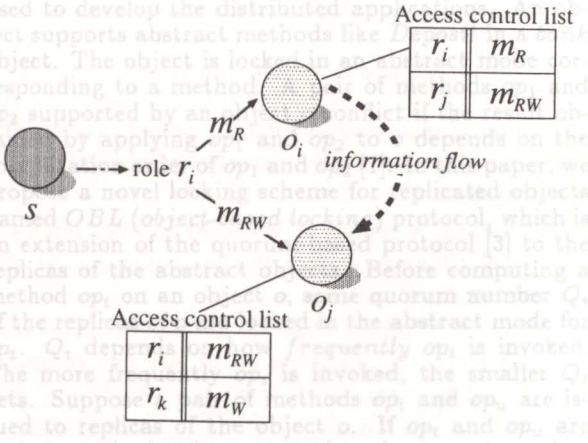


Figure 8. Information flow control.

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 performed on which object. Furthermore, we have discussed how to control information flow to occur through roles.

References

[1] Bell, D. E. and LaPadula, L. J., "Secure Computer Systems: Mathematical Foundations and Model," *Mitre Corp. Report,* No. M74–244, *Bedford, Mass.,* 1975.

[2] Bertino, E. and Martino, L., "Object-Oriented Database Management Systems : Concepts and Issues," *IEEE Computer,* Vol. 24, No. 4, 1991, pp. 33–47.

[3] Castano, S., Fugini, M., Matella, G., and Samarati, P., "Database Security," Addison-Wesley, 1995.

[4] Denning, D. E., "A Lattice Model of Secure Information Flow," *Communications of the ACM,* Vol. 19, No. 5, 1976, pp. 236–243.

[5] Denning, D. E. and Denning, P. J., *Cryptography and Data Security,* Addison-Wesley, 1982.

[6] Ferrai, E., Samarati, P., Bertino, E., and Jajodia, S., "Providing Flexibility in Information Flow Control for Object-Oriented Systems," *Proc. of 1997 IEEE Symp. on Security and Privacy,* 1997, pp. 130–140.

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

[8] Harrison, M. A., Ruzzo, W. L., and Ullman, J. D., "Protection in Operating Systems," *Communication of the ACM,* Vol. 19, No. 8, 1976, pp. 461–471.

[9] Gosling, J. and McGilton, H., "The Java Language Environment," Sun Microsystems, Inc, 1996.

[10] Lampson, B. W., "Protection," *Proc. of 5th Princeton Symp. on Information Sciences and Systems,* 1971, pp. 437–443. (also in *ACM Operating Systems Review,* Vol. 8, No. 1, 1974, pp. 18–24.)

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

[12] Object Management Group Inc., " The Common Object Request Broker : Architecture and Specification," Rev. 2.1, 1997.

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

[14] 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.

[15] Sybase, Inc., "Sybase Adaptive Server Enterprise Security Administration," 1997.

[16] 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.

[17] Tari, Z. and Chan, S. W., "A Role-Based Access Control for Intranet Security," *IEEE Internet Computing,* Vol. 1, No. 5, 1997, pp. 24–34.

[18] 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.