# Permission Based Granular Access Control Pattern

RUSS RUBIS, Florida Atlantic University
IONUT CARDEI, Florida Atlantic University

Enterprise applications are designed to address specific business needs and are generally run within the internal corporate networks. Access to enterprise applications is controlled by various corporate policies, based on numerous widely accepted patterns and frameworks. There has been a great amount of work performed in the area of Access Control models, ranging from simple MAC (Mandatory AC) and DAC (Discretionary AC) [11] models to more elaborate RBAC (Role-based AC) [9], and advanced SAC (Semantic AC) [10] and ABAC (Attribute-based AC) [12]. These models have been widely used in enterprise applications. One of the shortcomings of the access control models is that they are applied to protected objects as a whole, and do not provide a granular control over individual parts of these objects. Furthermore, most models do not address the type of access control being granted to the subject (or caller). For example, a control access model might provide read, write, and delete access on a protected object, but does not provide a way to protect individual parts of the said object. Often in enterprise applications there is a need to provide access to certain parts of the protected object, and to prevent access to other parts of the same object. In this paper we propose a pattern for permission based granular access control to protected objects within a given business application.

General Terms: Enterprise Applications

Additional Key Words and Phrases: Business Objects, Enterprise Applications, Access Control, Granular Access Control, Roles, Permissions, Groups

## 1. INTRODUCTION AND OVERVIEW

Many Access Control models address global access to protected objects. For example, once an access is granted to a protected object, the subject (user via an application or process via integration) has a full access to the object. A classic example is the bank customer being granted access to his or her bank account. In most models, once the customer has been identified, that customer has full access to his/her account, and is allowed to perform deposits, withdrawals, transfers, etc. The problem with the classic bank customer example is that the customer's activities are limited in scope and are oversimplified in the context of the example. In reality, enterprise applications are much more complex and require a more dynamic approach to defining the access control for the given user/process, and which parts of the protected object are under the said control and which ones are outside of it.

In this paper we propose a pattern for permission based granular access control to protected objects within a given business application. An enterprise application is comprised of one or more modules, which in turn are a collection of related business objects. For example, a Procurement Module is part of an Enterprise Resource Planning (ERP) application, and contains a set of business objects which support its features and processes. Following are examples of some of the business objects in a typical ERP procurement module:

- Requisition
- Purchase Order
- Invoice
- Vendor

This paper presents a pattern for a granular access control to business object in the context of a typical enterprise application.

## 2. PERMISSION BASED GRANULAR ACCESS CONTROL PATTERN

### 2.1 Intent

A holistic access control to business objects is not sufficient as business objects are often comprised of multiple parts (attributes, operations) and may contain references to other business objects. Objects need to be designed with built-in support for granular access control. Consequently, we need a model which provides a granular control to business objects. In this pattern we present a basic approach for access control to objects, attributes, and operations in environments with a manageable number of subjects.

### 2.2 Example

Consider a simple customer business object, which consists of customer name, address, telephone, email, credit card info, and ordering history.

The customer data should only be available to individuals who need it and have the right (permission) to access it. Most of the time, only parts of the customer's data is needed by those who access it. For example, a marketing department might need to see the customer's order history and contact info, but they should not be able to see the customer's credit card info. On the other hand, the ordering department should have access to the customer's credit card info in order to be able to validate it, but might not have a valid business reason to see the order history.

Access to the customer object and its attributes could be programmed into the object itself. This approach however is not desirable as it does not take into account constantly changing business conditions and processes. Any changes to the way customer object is accessed and controlled would require additional programming, which is often expensive and time consuming.

An alternative is to associate permissions with the customer object and its attributes/operations. Thus, for example, a marketing department permission can be associated with the customer business object, and its contact info and ordering history attributes. At the same time, an ordering department permission can be associated with the customer business object's credit card attribute. Both permissions are associated with the customer object, but each propagates to a different set of its attributes, thus each permission provides a granular access control over the customer object.

### 2.3 Context

You are designing a new business object and need a way to provide granular access control to the new object and its attributes/methods when your business object is deployed in your business application. The granular access control should be configurable and verifiable. Finally, the granular access to the business object should also provide control over the business object's attributes' visibility (read access) and editability (update access).

### 2.4 Problem

Access to business objects, their attributes and operations must be given individually to subjects in a granular way. A subject may be given access to an object, but not to all its attributes and operations. Also, access to Read, Update, Copy, Create, and Delete must be differentiated – per attribute. The system must apply policies to deal with situations where a subject's rights for an object conflicts with the specific access rights of that subject for the required attribute/operation. For instance, a subject that has the Read access right for an object may lack the Read right for one or more of its attributes (i.e. credit card number on the customer object).

### 2.5 Solution

The design in Figure 1 assigns RightsDescriptor objects to a subject that describe specific access type allowed for the subject to individual objects, their attributes, and operations. An optional RightsDescriptor object is assigned to each business object, attribute, and operation to describe access rights that apply for all subjects that request access. The PermissionAuthority object validates access by matching the rights a subject has for a specific object, attribute, or operation with the (optional) rights descriptor owned by corresponding object, attribute, or operation. Validation must consider cases when a subject does not have access rights objects for the requested access and when the object (or attribute/operation) has no rights descriptor. Best practices (assigning least privileges) can be applied.

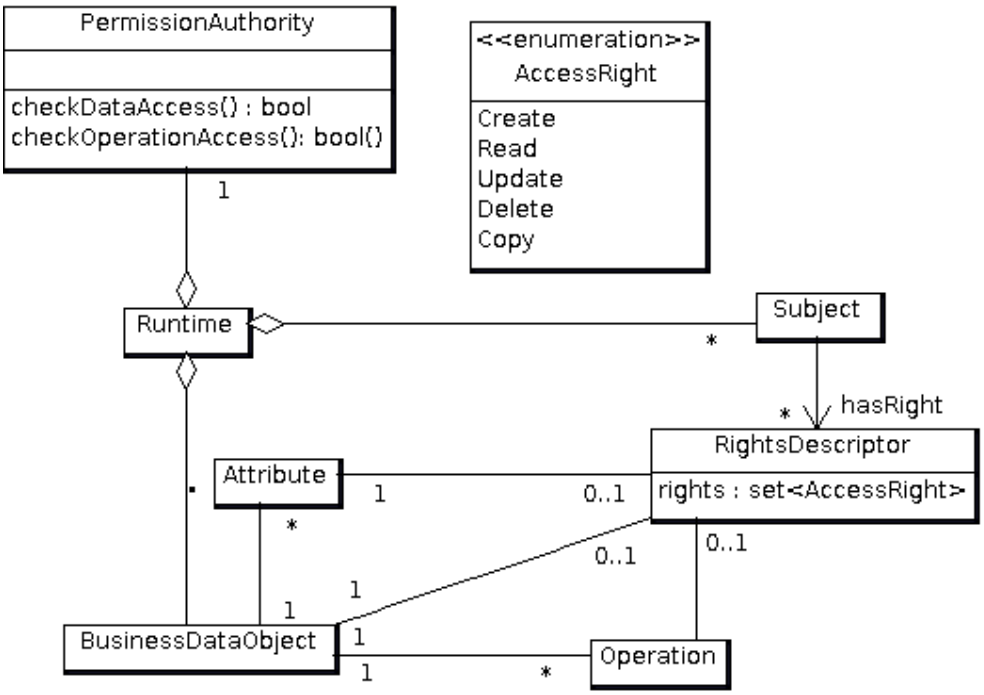


**Figure 1: The class diagram for the Permission Based Granular Access Control pattern.**

All object level permissions are propagated by to the attributes and operations of the object. At the attribute and operation level, these permissions can be overridden by explicitly assigning another permission to an attribute or operation.

Figure 2 shows the sequence diagram granting access to business objects and their attributes/operations using the Permission Based Granular Access Control pattern. When subject requests a specific access to a business object, we first must verify that at least one of the permissions assigned to the subject matches the permission required to perform the access requested. If the object level match is successful, we can then perform granular matches on the objects individual attributes and operations. Ones these matches are performed, the subject will either get nothing (permission match failed on the requested business object), or the subject will get access to the requested business object and those attributes/operations which passed granular permission match.

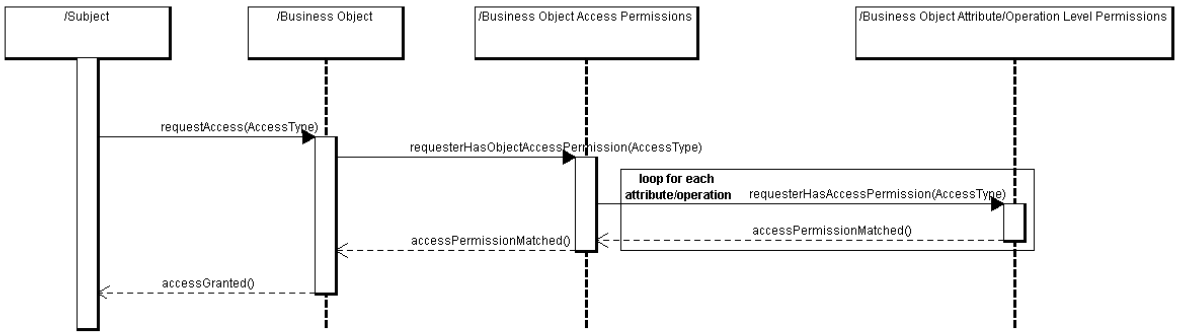


**Figure 2: The sequence diagram for Permission Based Granular Access Control pattern.**

The following table provides an example of how permissions can be mapped to a business object and its attributes. We use "/" to denote object/attribute relationship. For example, a customer object with its attribute credit card would be denoted as Customer/CreditCard.

**Table 1: Permission based granular attribute access.**

| | Customer Object | Customer/CreditCard Attribute | Customer/Telephone Attribute |
|---|---|---|---|
| Create Permission | CustomerService | | |
| Read Permission | CustomerService or Finance | Finance | |
| Update Permission | CustomerService or Finance | Finance | |
| Delete Permission | Finance | | |
| Copy Permission | CustomerService | | |

Above table shows permission mappings for Customer business object and some of its attributes. Subjects with CustomerService permission are granted access to create, read, update, and copy a Customer business object. However, only users with Finance permission can delete a Customer business object. Because Customer/Telephone field does not have any permissions assigned to it explicitly, it inherits the access permissions of the object itself. The attribute Customer/CreaditCard, on the other hand, can only be updated or viewed by subjects with Finance permission. Consequently, subjects with CustomerService permission may enter customer credit card info at the time of Customer business object instance creation, but do not have access to update or view the credit card info subsequently.

Thus via granular assignment of permissions we can provide a granular access control to a business object as a whole, as well as to its individual attributes and operations.

### 2.6 Consequences

This pattern is suitable for systems where it is feasible to assign all subjects rights descriptors for objects they would access, including for their attributes and operations. For each object (and each of its attributes/operations) it only requires optional specification for just one rights descriptor that applies to all subjects. An advantage of this approach is that the permissions associated with the given business object access control do not have to be determined at design or development time. Rather, the mapping of permissions to objects can be performed at run-time, after the business object has been deployed in the business application. Assigning permissions at run-time also enables us to group the business objects based on the permission(s) to which they are mapped. Because we can group permissions into roles, we can easily incorporate our configuration into an existing Role Based Access Control (RBAC) type architectures.

## 3. RELATED PATTERNS AND FRAMEWORKS

### 3.1 Discretionary Access Control (DAC)

The DAC pattern enforces access control based on user identities and the ownership of objects. The owner of an object may grant permission to another user to access the object, and the granted user may further delegate the per-mission to a third person. [11]

### 3.2 Mandatory Access Control (MAC)

The MAC pattern governs access based on the security level of subjects (e.g., users) and objects (e.g., data). Access to an object is granted only if the security levels of the subject and the object satisfy certain

constraints. The MAC pattern is also known as multilevel security model and lattice-based access control. [11]

### 3.3 Role Based Access Control (RBAC)

The RBAC pattern enforces access control based on roles. A role is given a set of permissions, and the users assigned to the role acquires the permissions given to the role. Since the RBAC pattern is based on roles which are in general fewer than the number of users, it is useful for managing a large number of users. [9]

### 3.4 Attribute-based Access Control (ABAC)

Attribute-based access control defines an access control architecture whereby access rights are granted to users through the use of policies which combine attributes together. The policies can then use any type of attributes (user attributes, resource attribute, etc...). Attributes can be compared to static values or to one another thus enabling relation-based access control. [12]

### 3.5 Semantic Access Control (SAC)

The Semantic Access control model was created in 2002. The fundamentals of this semantics-based access control model are the definition of several metadata models at different layers of the Semantic Web. Each component of SAC represents the semantic model of a component of the access control system. The semantic properties contained in the different metadata models are used for the specification of access control criteria, dynamic policy allocation, parameter instantiation and policy validation processes. [13]

## 4. CONCLUSIONS AND FUTURE WORK

The permission based granular access control to common business objects and their entities provides a clear, generic and extendible design-time approach to securing access to business data. It allows for use of as many or as granular permissions to control access to various parts of business data in a declarative way.

It is important to note that the permissions are independent of the access control model used. The content of the permission-specific attributes can be adapted to each access control model. For instance, in a Mandatory AC model the content of a permission attribute would be the security level, in the Discretionary AC model it would point to specific users, in RBAC it would contain roles, and finally in the Semantic AC model, it would point to AC policies.

## 5. ACKNOWLEDGMENTS

We would like to thank Antonio Maña for his invaluable feedback and support during the shepherding of this paper.

## 6. REFERENCES

[1] A Guide To The Sarbanes-Oxley Act, The Sarbanes-Oxley Act of 2002, DOI=http://www.soxlaw.com/
[2] Fowler M., 1997. Analysis Patterns: Reusable Object Models, Addison-Wesley Longman.
[3] Daum B., 2003. Modeling Business Objecs with XML Schema, Morgan Kaufmann Publishers.
[4] Eriksson H.-E., Penker M., 2000, Business Modeling with UML: Business Patterns at Work, OMG Press / Wiley Computer Publishing
[5] Adams J., Koushik S., Vasudeva G., Calambos G., 2002, Patterns for e-business: A Strategy for Reuse, IBM Press
[6] Buckl S., Matthes F., Monahov I., Roth S., Schulz C., Schweda C.M., 2012. Enterprise Architecture Management Patterns for Company-wide Access Views on Business Objects, ACM Transactions on Applied Perception, Vol. 2, No. 3, Article 1, Publication date: May 2012.
[7] Chang K.Y., Chen L.S., Lai C.K., 1999. Document-View-Presentation Pattern, Department of Electrical Engineering, National Cheng-Kung University, Taiwan.
[8] E. B. Fernandez and Pan R. A Pattern Language for Security Models. In Proceedings of the 8th Conference on Pattern Language of Programs (PloP), Monticello, IL, 2001.
[9] D. Ferraiolo, D. R. Kuhn, and R. Chandramouli. Role-Based Access Control. Artech House, 2003.

[10] Yagüe, M. I., & Maña, A. (2005). Semantic access control model: A formal specification. In Computer Security–ESORICS 2005 (pp. 24-43). Springer Berlin Heidelberg.
[11] Kim, D. K., Mehta, P., & Gokhale, P. (2006, October). Describing access control models as design patterns using roles. In Proceedings of the 2006 conference on Pattern languages of programs (p. 11). ACM.
[12] Yuan, E., & Tong, J. (2005, July). Attributed based access control (ABAC) for web services. In Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on. IEEE.
[13] Semantic Access Control, A Semantics-based Access Control Model for Open and Distributed Environments, http://www.lcc.uma.es/~yague/Semantics-basedAccessControl.html