TDDC 03 Project: Spring 2004

An Evaluation of RBAC Policy Languages for Web Applications

Jing Zhang
Xiaojie Shen

Supervisor: Almut Herzog

An Evaluation of RBAC Policy Languages for Web Applications

Jing Zhang
Institute of Technology
Linkoping University
SE-581 83 Linkoping, Sweden
jinzh122@student.liu.se

Xiaojie Shen
Institute of Technology
Linkoping University
SE-581 83 Linkoping, Sweden
xiash506@student.liu.se

ABSTRACT
The rapid growth of the Internet and a range of web applications bring the urgency of security issues, especially for access control. Role-based Access Control (RBAC) is recognized as a superior alternative and less error-prone to traditional discretionary and mandatory access controls. In this paper, we examine the representation of RBAC policies in web applications under distributed environments. Firstly, several important requirements and features for RBAC policy languages, especially with the consideration of web applications are identified. They are expressive, inter-operable, applicable to heterogeneity, flexible, manageable, and efficient. Then we categorized the existing RBAC policy languages into four categories: XML-based, UML-based, Object-oriented programming languages, and Constraint logic languages. Each category is carefully examined and evaluated, and a comparison with respect to the requirements is given. We conclude with recommendations for XML as a basis for a RBAC policy language.

Keywords
Policy language, RBAC, Web applications

1. INTRODUCTION
The rapid diffusion of the Internet and the growth of its key enabling technologies are producing a significant growth of the demand and unprecedented opportunities for web applications. Security issue is getting more concerns. Particularly there are great demands on access control services, which need to be deployed in interconnected, and interactive environment. Users of the system must be authenticated to be legitimate users, and must only be permitted to retrieve and modify data in the ways that are authorized by an access control policy.

Role-Based Access Control (RBAC) has emerged as a proven and superior alternative to traditional discretionary and mandatory access controls. RBAC greatly simplifies the management of permissions by associating them to which users are assigned, thereby acquiring the roles' permissions [1]. Another advantage of RBAC is being "policy-neutral", which means that a sophisticated RBAC-service may be configured to enforce many different access control polices including DAC- or MAC-based policies [2]. RBAC is receiving constant interest in both research and industry, but most of the work is about RBAC models and frameworks, or implementations. Little attention is given to managing and expressing access control policies, especially for web applications in distributed environment. For web applications, beyond providing strong protection, security systems must also be flexible and promote inter-operability between different domains of trusts under distributed environments. The nature of web applications like transaction and activity intensive, written in a variety of languages, running in different operating systems, and rapid changes determines the security system must be efficient, manageable, and flexible. There are special requirements for policy control for web applications. The purpose of this paper is to examine and evaluate existing policy languages for RBAC in web applications.

2. ROLE-BASED ACCESS CONTROL (RBAC)
Role-Based Access Control (RBAC), a policy neutral access control mechanism, is widely known as being an inherently easier and less error-prone way of administrating access control policies, as compared to traditional discretionary and mandatory access controls. The basic principle of RBAC is the separation of permission assignments (PA) and user assignments (UA) [3]. With RBAC, permissions are assigned to roles and roles are assigned to users. A user thereby acquires the permissions assigned to that specific role. As roles represent organizational responsibilities and functions, a role-based model directly supports arbitrary, organization-specific security policies. Since permissions are de-coupled from users, changes to permission or user assignments have minimal isolated impact on administration. Within an organization the description of roles tend to change significantly slower than the assignment of individuals to these roles.

The RBAC security model is only an abstract and general model; there are many interpretations of it and it is a mechanism that can implement a variety of policies. The central concepts of RBAC are users, roles and permissions. RBAC policies such as Role assignment, Role authorization, Permission assignment, Role hierarchy, constraints, need to be represented in security systems. RBAC policy language is the language used to represent access control policies and to express constraints. Currently there is no standard for RBAC policy language, people implement RBAC in different ways using different languages.

3. REQUIREMENTS FOR RBAC POLICY LANGUAGES IN WEB APPLICATIONS
For web applications in distributed environments, beyond providing strong protection, the security systems require more features, thus there are more requirements for policy languages. We have identified the following important features:

Expressive. As a policy language, one general requirement is expressiveness, that is, it can express or represent the policies clearly and precisely. Although it is not specific for RBAC policy languages in web applications, we include it here as the basic requirement or prerequisite when we evaluate a policy language.

Inter-operable. In web applications, usually under distributed environments, authorization often requires cooperation among separate, autonomous administrative domains. Maintaining a consistent authorization strategy requires each system to maintain at least some knowledge of its potential collaborators throughout the entire system. Also any authorization decisions that span two or more domains require the coordination of all participants [5]. Thus the policy languages should be inter-operable, able to work across domains.

Applicable to Heterogeneity. The distributed environment means heterogeneous environments, with different operating systems, different management tools. The web applications can be written in a variety of languages. The policy languages need to be applicable to these environments.

Flexible. In nowadays' information systems, especially in web applications, changes happen very frequently. The changes can mean access control policies, or technologies in use, or security mechanisms. So the languages need to be flexible enough to adapt to these changes.

Manageable. Often in a large networked systems, the security administration is very complicated, costly and error prone. It indicates that the policy languages should be manageable for administration.

Efficient. In the web application environments, the activities and transactions are very intensive. The process of authenticating and authorizing needs to be fast. The policy languages are desirable to be efficient.

4. POLICY LANGUAGES

This section is an evaluation of existing policy languages. Currently there are policy languages for different developing environments, for different applications, and for different operating systems. We roughly classify them into four categories: XML-based, UML-based, Object-oriented programming languages, and Constraint logic programming languages.

4.1 XML-based

As a document markup language, XML (eXtensible Markup Language) is being widely used on the Internet or in web applications. There are also quite a lot of work and research being done to exploit XML as a security policy language, in RBAC for example. Vuong, Smith and Deng [3] present their work in using XML to implement RBAC policy. Chandramouli [8] describes an application of using XML to represent the RBAC policy.

XML is designed as a meta-language for Internet use. Its objectives are to overcome the rigid HTML tagging scheme while providing web users with a means for defining their own domain specific tags and attributes. In implementing RBAC, XML Document Type Definition (DTD) is used for representing the schema of a RBAC model and a conforming XML document will contain the actual RBAC-based access control data. Then normally some Java program is developed to read the data in the XML documents.

For example, Vuong, Smith and Deng [3] model each RBAC component as an XML element:

A *User* is represented as

```
<!ELEMENT USER EMPTY>
<!ATTLIST USER NAME ID #REQUIRED>
```

The above syntax defines a new XML tag of type USER with a required NAME attribute of type ID that by default is unique.

A *Role* is represented as

```
<!ELEMENT ROLE EMPTY>
<!ATTLIST ROLE TITLE ID #REQUIRED>
```

A *Permission* is implementation-specific; therefore, it is modeled as an abstract representation that requires definition when defining policies. A Permission is represented as

```
<!ELEMENT PERMISSION EMPTY>
<!ATTLIST PERMISSION %DEFINITION;>
```

A *Permission Assignment* assigns a set of permissions to a role; it is represented as

```
<!ELEMENT PERMISSION_ASSIGNMENT EMPTY>
<!ATTLIST PERMISSION_ASSIGNMENT
          ROLE IDREF #REQUIRED
          PERMISSIONS IDREFS #REQUIRED>
```

A *Role Assignment* assigns a set of users to a role; it is represented as

```
<!ELEMENT ROLE_ASSIGNMENT EMPTY>
<!ATTLIST ROLE_ASSIGNMENT ROLE IDREF
          #REQUIRED
          USERS IDREFS #REQUIRED>
```

A *Role Hierarchy* is represented as a set of INHERITS elements, each of which associates a set of junior roles to a senior role:

```
<!ELEMENT INHERITS EMPTY>
<!ATTLIST INHERITS FROM IDREFS #REQUIRED
          TO IDREF #REQUIRED>
```

With the defined RBAC components as XML elements, then an instance of an RBAC model is the composition of various RBAC components. For example, an RBAC1 security model is represented as a production rule.

```
<!ELEMENT RBAC1_MODEL (USER+, ROLE+,
INHERITS*,PERMISSION+,
PERMISSION_ASSIGNMENT*, ROLE_ASSIGNMENT*)>
```

A sample XML representation of a hypothetical RBAC policy is given as follows:

```
<?xml version="1.0" encoding="UTF-8"
standalone="no" ?>
<!DOCTYPE RBAC1_MODEL SYSTEM
"htpp://www.cs.fiu.edu/~nvuong01/RBAC1_MODEL
.dtd">
<RBAC1_MODEL TYPE_NAME="RBAC1_POLICY">
<!-- User set definition -->
<USER NAME="a"></USER>
```

```
<USER NAME="b"></USER>
...
<!-- Role set definition -->
<ROLE TITLE="Caregiver"></ROLE>
<ROLE TITLE="Nurse"></ROLE>
...
<!-- Role hierarchy definition -->
<INHERITS FROM="Caregiver"
TO="Registrar"></INHERITS>
<INHERITS FROM="Caregiver"
TO="Nurse"></INHERITS>
...
<!-- Permission set definition -->
<PERMISSION PERMID="P1"
OPERATION="RW" RESOURCE="AMD">
</PERMISSION>
<PERMISSION OPERATION="R" PERMID="P2"
RESOURCE="PST"></PERMISSION>
...
<!-- Permission assignment -->
<PERMISSION_ASSIGNMENT
ROLE="Psychiatrist"
PERMISSIONS="P1">
</PERMISSION_ASSIGNMENT>
<PERMISSION_ASSIGNMENT
PERMISSIONS="P2 P4 P6 P10 P11"
ROLE="Physician">
...
<!-- Role assignment -->
<ROLE_ASSIGNMENT ROLE="Psychiatrist"
USERS="a"></ROLE_ASSIGNMENT>
<ROLE_ASSIGNMENT ROLE="Technician"
USERS="d f"></ROLE_ASSIGNMENT>
...
</RBAC1_MODEL>
```

XML, as a structured language, is very expressive and flexible. And if properly designed, a structured language is closer to natural language than any other method for representing security policies [3], thus makes it more readable. It provides a very accessible notation for expressing the semantics of RBAC policies. Also XML is a platform independent data exchange format; combined with Java program, it has no problem with different platforms and inter-operability. So it is possible for web applications on multiple servers under distributed environment. Depending on different DTDs, the same XML document can be applied to different models or schemas. So it is quite flexible. With the proliferation of XML in the industry, there is a high probability that future systems will be equipped with an XML parser.

Although XML format is very simple, the writing and maintaining of the documents are very tedious. Currently there are no standard and satisfied tools for it. So the administration work is quite error prone and difficult. This can be a drawback for XML as a policy language used for an enterprise that requires large and complex security policies.

On top of XML, OASIS ratified XACML (eXtensible Access Control Markup Language), a standard, general-purpose access control policy language [5]. It was designed to accommodate most system needs, so it may serve as a single interface to policies for multiple applications and environments. In [9], PERMIS provides a Privilege Management Infrastructure (PMI), uses X.509 Attribute Certificates to specify subject attributes such as roles and permissions. The RBAC policy is in XML format to control access to all the targets within the policy domain and is composed of a number of sub-policies. The PERMIS project is currently investigating the use of XACML as a core language to replace parts of their proprietary policy language. XACML, as the standard, can be very promising to support RBAC in distributed systems under heterogeneous environments.

4.2 UML-based

In recent years, UML (Unified Modeling Language) has emerged as a de facto standard for object-oriented modeling of software intensive systems. It is also being used as an approach to represent RBAC policies.

Basin, Doser and Lodderstedt [6] propose the UML-based security modeling language SecureUML for modeling access control requirements that generalized RBAC. Figure 1 presents the metamodel that defines the abstract syntax of SecureUML. The types *User*, *Role*, and *Permission* and the relations *UserAssignment*, *PermissionAssignment*, and *RoleHierarchy* are directly adopted from RBAC standard.

There are also some additions. An *AuthorizationConstraint* is a logical predicate that is attached to a permission by the association *ConstraintAssignment* and makes the permission's validity a function of the system state. The types *Resource* and *Action* roughly correspond to the terms *Operation* and *Object*. Each resource offers one or more actions and each action belongs to exactly one resource, which is denoted by the composite aggregation *ResourceAction*. There are two categories of actions *AtomicAction* and *CompositeAction*. Atomic actions are low-level actions that can be mapped directly to actions of the target platform. While composite actions are high-level actions that may not have direct counterparts on the target platform. Additional constraints are given using expressions in the Object Constraint Language (OCL). (OCL is used to specify constraints on objects in UML. It has the power but not syntax of the Lower order Predicate calculus plus simple set theory.)

An example policy is shown as the Figure 2. *Member* and *GoldMemeber* are the roles, they have several permissions to the process Ordering. The up left corner is the constraint.

Epstein and Sandhu [10] also present the usage of UML to define RBAC models in an existing RBAC Framework for Network Enterprises (FNE). It is quite similar to the above, only they use a different UML notation and document application constraints as preconditions in plain English, whereas in SecureUML, OCL is used.

In most cases, UML is able to represent an RBAC cleanly. It can be expressive enough to naturally and concisely describe complex security policies. Compared to other languages, UML is more straightforward and visually clear. Also if the UML generator can

create more detailed code, it can save programming time. It enhances portability since models are technology independent and hence the migration to new technologies can be realized by changing the generation rules, not the models themselves. In a wider view, it integrates security models with UML process models, which closes the gap between software engineering and security engineering. This means that security can be tightly integrated into a system during design, rather than after-the-fact, increasing the security and maintainability of the resulting system [6].

But there are some weaknesses in using UML to document RBAC models [10]. The expressiveness of the language still has some room for improvements. Although there is a check on the UML syntax, there is no logic or semantic check. It is solely the designers' responsibility to accurately depict the model.

4.3 Object-Oriented Programming Language

In the last few years, Java is becoming one of the most popular programming languages. Its advantages like platform independent, object-oriented have made it quite successful in web applications. Besides its own security features and services, Java is also explored to implement other security policies like RBAC policies [11] [12].

Giuri [11] [12] presented a JRBAC-99 (Java RBAC) model, which takes advantage of a Java security model in JDK 1.2 based on the concept of protection domain, and the JAAS (Java Authentication and Authorization Service) to implement RBAC policies for Java-based web applications. There are two new *Principal* implementations defined: *UserPrincipal* and *RolePrincipal*. The permission-role-assignment (PRA) and permission-user-assignment (PUA) relationships are directly implemented using the JAAS Policy. To implement the role-role-assignment (RRA) and the user-role-assignment (URA) relationships it is necessary to provide a new *RolePolicy* class, which can use a file represented with a syntax that is similar to the JAAS policy file syntax:

```
grant [role "role-name" | user "user-name"]
{
  role "role-namel" [default];
  . . .
  role "role-nameN" [default];
};
```

To manage role activation, the *RoleController* class provides the following methods:

- reset(): disables every role;
- resetDefaults(): disables every role and enables default roles only;
- enableRole(String roleName): adds the role identified by roleName to the set of enabled roles;
- enabledRoles(): retrieves the set of currently enabled roles.

The JRBAC-99 model also allows the specification of constraints on users and roles. For example, to specify activation constraints, the role policy file must be extended to accept the following syntax:

```
grant [role "role-name" | user "user-name"]
{
  role "role-namel" [default];
       constraint ConstraintClass "par1" …;
  . . .
  role "role-nameN" [default];
};
role "role-name"
       constraint ConstraintClass "par1" …;
user "user-name"
       constraint ConstraintClass "par1" …;
```

Direct RBAC support by the JDK would be very helpful for application developers, and it diminishes the need for proprietary extension. On the other hand, it only applies to Java applications. Although it is possible to add RBAC features to the current JDK, they are still limited to policies where it is sufficient that the set of permissions for a given CodeSource is fixed, and is statically computed at object creation time [11]. This also causes a proliferation of permission-related structures within the system, with a possible reduction of the overall system performance. The policies written in the code itself can cause many problems: inflexible to changes, not applicable to non-developers for administration.

We have also seen using of some other object-oriented programming languages to implement RBAC polices and models, such as Alloy language in [13], and XOTcl in [2]. They are used for different developing environments and different operating systems. By that we know it is possible to implement the policies in the programming languages, but the problems like Java we stated above still remain.

4.4 Constraint Logic Language

Another branch is using constraint logic languages to specify RBAC policies. Barker and Stuckey [15] show their use of CLP (Constraint Logic Programming) for formulating and implementing RBAC polices. Covington etc. [14] choose to use a prolog-style logical language for expressing polices. Ahn and Sandhu [7] introduce a formal language RCL 2000 (Role-based Constraint Language) for specifying role-based authorization constraints.

Take RCL 2000 for example, it has six entity sets called users (U), roles (R), objects (OBJ), operations (OP), permissions (P), and sessions (S). They are defined as follows:

-U = a set of users, {u1, …,un}

-R = a set of roles, {r1,…,rm}

-OP = a set of operations, {op1, …,opo}

-OBJ = a set of objects, {obj1,…,objr}

-P = OP x OBJ, a set of permissions, {p1,…, pq}

-S = a set of sessions, {s1,…,sr}

…

There are functions *user* gives us the user associated with a session and *roles* gives us the roles activated in a session. The user assignment relation UA is a many-to-many relation between users and roles. Similarly the permission-assignment relation PA is a many-to-many relation between permissions and roles. Users are

authorized to use the permissions of roles to which they are assigned.

Besides that, additional elements and system functions beyond the RBAC model are defined in RCL2000, CR (a collection of conflicting role sets), CP (a collection of conflicting permission sets), CU (a collection of conflicting user sets), and two nondeterministic functions, OE (oneelement) and AO (allother). The OE(X) function allows us to get one element xi from set X. And with AO(X) we can get a set by taking out one element. To illustrate how to use these functions to specify role-based constraints, here is an example: No user can be assigned to two conflicting roles. In other words, conflicting roles cannot have common users. It can be expressed as $|roles(OE(U)) \cap OE(CR)| \leqslant 1$. Similarly, all the other RBAC constraints can be easily specified.

Constraint logic languages are powerful in representing role-based authorization constraints, succinct and concise, strong technical results that enable properties of a policy to be proved, efficient in performance. But in practice, it would be frustrating and clumsy for a policy administrator to manage, especially when editing large, complex policy files. It also requires the administrator to have knowledge in logic languages and semantics. A user-friendly front-end to the language is more preferable to make the roles and policy rules more visual and easy-to-understand.

4.5 Comparison

After the examination and evaluation of the different policy languages above, we know how each language is used for specifying RBAC policies, and every one of them has some advantages and disadvantages. To get a better view, we have a comparison of these languages in Table 1, with respect to the requirements for policy languages of RBAC in web applications.

XML-based language is very expressive, the same for Object oriented languages and Constraint logic language, and they all can represent the policies and constraints precisely. While UML-based has its limitations in expressing and checking of constraints.

As for Inter-operable requirements, XML-based language works quite fine, and is becoming a standard for security policies in distributed environment. As for UML-based, it is possible to migrate models across domains. For Object oriented language and Constraint logic language, it is possible to inter operate, but it may has more requirements for different domains.

Applicable to heterogeneity: XML-based is platform independent, and it usually works with Java, so it is no problem here. For UML-based it is possible to migrate the models. For Object oriented language, probably only Java is good at this requirement. But Constraint logic language is usually depending on specific environments or operating systems.

Except for Object oriented language, which combine the policy inside the code itself, has the problem of inflexible to change, the other three categories are good at adapting to changes and quite flexible. Among them, XML-based is the best in flexibility. Depending on different DTDs, it can be applied to different models or schemas.

For a large web application, the access control policies are always complicated and difficult to manage. UML-based is very good at this requirement, and it is more straightforward and visually clear. XML-based is also quite good, with easy semantics. But if the policies are huge, it is preferable to have some tools to manage XML documents. But as for Object-oriented language, it requires programming knowledge, which only is applicable to developers; and for Constraint logic language, it is also not practical for administrators to manage, since it requires the knowledge of logic language.

And finally efficiency, here the best one is Constraint logic language, which has the mathematics foundation and strong technical properties.

From the comparison, we can see the XML-based policy language has fulfilled almost all of the requirements, while others are good at some aspects in some situations. Thus we consider XML is a very promising language for RBAC policy in web applications.

5. CONCLUSION

The Internet provides the opportunities to a range of web applications. The access control issue is still a hot topic. As a proven and superior alternative to traditional discretionary and mandatory access controls, RBAC has many of its advantages. There are many tries in both industry and research to implement it in web applications and under distributed environments. In this paper we are interested in how the RBAC policies can be represented in web applications.

First, we identified several important requirements and features for RBAC policy languages, especially with the consideration of web applications. Then we examined different types of existing RBAC policy languages, and classified them into four categories: XML-based, UML-based, Object-oriented programming languages, and Constrain logic languages. Evaluations of each category are given and followed a comparison with respect to the requirements we stated before. Compared to others, XML shows its advantages in aspects like interoperable and manageable. We consider it as a basis for RBAC policy language.

We realize, however, our search of policy languages is not exhaustive, and the comparisons are not complete in every aspect. For different situations, under different environments, some policy languages will be better than others. So our recommendation is only for general purpose in most cases. And most likely there will emerge some new policy languages, which is more suitable and superior. We will be looking forward to that.

6. ACKNOWLEDGMENTS

Here we would like to give our deepest gratefulness to our advisor, for her valuable advice and support.

7. REFERENCES

[1] Al-Kahtani M., Sandhu R., "Access Control Models and Mechanisms: Induced role hierarchies with attribute-based RBAC", Proceedings of the eighth ACM symposium on Access control models and technologies, July 2-3 2003, pp142-148

[2] Neumann G., Strembeck M., "Access Control: Design and implementation of a flexible RBAC-service in an object-oriented scripting language", Proceedings of the 8th ACM conference on Computer and Communications Security, November 5-8, 2001, pp58-67

[3] Vuong N., Smith G., Deng Y., "Managing security policies in a distributed environment using eXtensible markup language

(XML)", Proceedings of the 2001 ACM symposium on Applied computing, March 2001, pp405-411

[4] Sandhu R., Coyne E., Feinstein H., Youman C., "Role-Based Access Control Models", IEEE Computer, 29(2), pp38-47

[5] Lorch M., Proctor S., Lepro R., Kafura D., Shah S., "Access control: First experiences using XACML for access control in distributed systems", Proceedings of the 2003 ACM workshop on XML security, October 2003, pp25-37

[6] Basin D., Doser J., Lodderstedt T., "RBAC for Collaborative Environments: Model driven security for process-oriented systems", Proceedings of the eighth ACM symposium on Access control models and technologies, June 2-3 2003, pp100-109

[7] Ahn G., Sandhu R., "Role-based authorization constraints specification", ACM Transactions on Information and System Security (TISSEC), November 2000, Volume 3 Issue 4, pp207-226

[8] Chandramouli R., "Application of XML tools for enterprise-wide RBAC implementation tasks", Proceedings of the fifth ACM workshop on Role-based access control, July 2000, pp11-18

[9] Chadwick D., Otenko A., "Applications: The PERMIS X.509 role based privilege management infrastructure", Proceedings of the seventh ACM symposium on Access control models and technologies, June3-4 2002, pp135-140

[10] Epstein P., Sandhu R., "Towards a UML based approach to role engineering", Proceedings of the fourth ACM workshop on Role-based access control, October 1999, pp135-143

[11] Giuri L., "Role-based access control on the Web using Java", Proceedings of the fourth ACM workshop on Role-based access control, October 1999, pp11-18

[12] Giuri L., "Role-based access control in Java", Proceedings of the third ACM workshop on Role-based access control, October 1998, pp91-100

[13] Schaad A., Moffett J., "Access Control Policies and Specifications: A lightweight approach to specification and analysis of role-based access control extensions", Proceedings of the seventh ACM symposium on Access control models and technologies, June 2002, pp13-22

[14] Covington M., Long W., Srinivasan S., Dev A., Ahamad M., Abowd G., "Securing context-aware applications using environment roles", Proceedings of the sixth ACM symposium on Access control models and technologies, May3-4 2001, pp10-2

[15] Barker S., Stuckey P., "Flexible access control policy specification with constraint logic programming", ACM Transactions on Information and System Security (TISSEC), Volume 6, Issue 4, November 2003, pp501-546

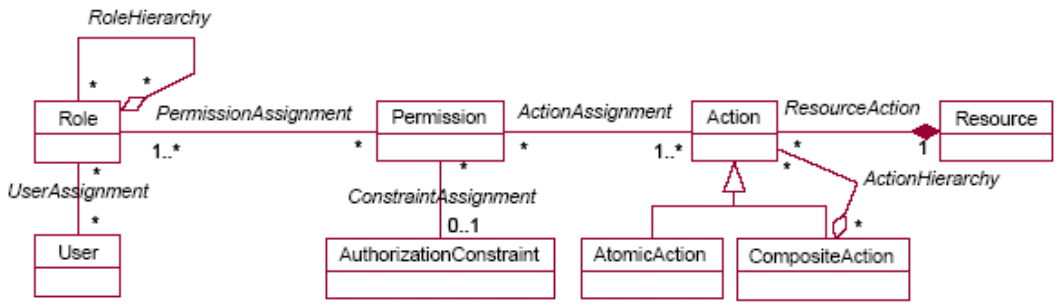

Figure 1 SecureUML metamodel

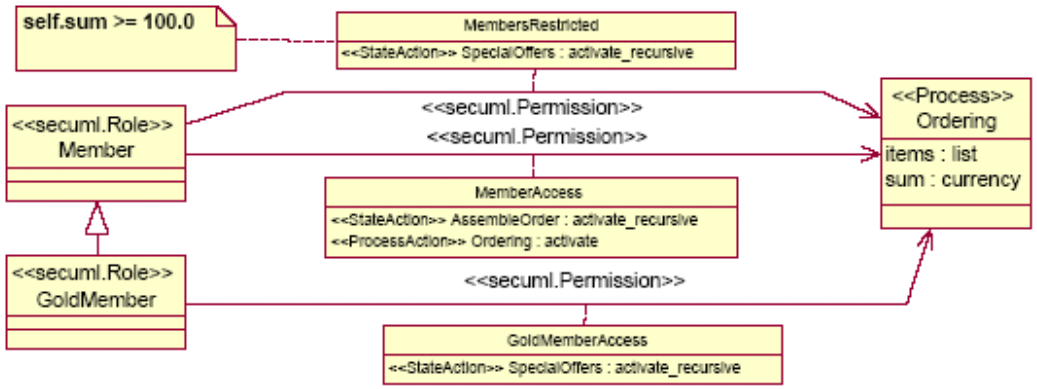

Figure 2 An Access Control Policy Example

	XML-based	**UML-based**	**Object-oriented programming languages**	**Constraint logic languages**
Expressive	Yes	Not very good	Yes	Yes
Inter-operable	Yes	Yes	Not very good	Not very good
Applicable to heterogeneity	Yes	Can be supported	Yes for Java	Probably not
Flexible	Yes	Yes	No	Yes
Manageable	Yes, but tools are preferred	Good	Applicable only to developers	Require knowledge of logic language
Efficient	Yes	Not very good	Not very good	Yes

Table 1 Comparison of policy languages