Applying Model Driven Architecture approach to Model Role Based Access Control System

By
Xin Jin

Thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

**Master of Science
in
System Science**

Under the auspices of the System Science Program

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Ottawa, Ontario, Canada

Abstract

The Role Based Access Control (RBAC) Model is used widely in distributed network systems to protect shared data and information. It uses security specifications to control the access to critical resources and avoid security violations. However, the security specifications are often defined as an ad-hoc manner in the deployment stage after the software design. This causes increasing vulnerabilities to the IT systems. Moreover, the security specifications are normally documented in files. Hence, a tool to support the security specifications editing and generation can increase the productivity and the quality of the security specifications.

To address the above problems, we present a tool-supported framework which uses the Model Driven Architecture (MDA) approach with the Unified Modeling Language (UML) profile to build the RBAC applications and security specifications for distributed systems. We illustrate how a user can use the UML profile that is presented in this paper to design security specifications for RBAC system at the beginning of the software design phase, carry it through the entire development process, and automatically generate system security specifications in eXtensible Access Control Markup Language (XACML) format. This methodology would enable software developers to become more productive and maintain high standards of software quality.

Acknowledgements

First, I would like to thank my supervisor Dr. Alan Williams for his technical and financial support for this work. Without his detailed reviews and feedback on every chapter, this thesis would never be finished. I also appreciate his support for allowing me to attend the Internship program at IBM in the summer of 2005, which was a great learning experience for me.

I would also like to thank the members of the RDA Research Group and ASERT Research Group during this research. The variety of presentations in these two groups has broadened my view in my work.

Finally, I want to thank my parents and my husband for their understanding and support. I especially would like to thank my son, KaiZhe, who always plays by himself to give me more time in study.

Table of Contents

Index of Figures

Index of Tables

List of Acronyms

Acronym	Definition
ANSI	American National Standards Institute
CASE	Computer-Aided Software Engineering
DSD	Dynamic Separation of Duty
EMF	Eclipse Modeling Framework
EMOF	Essential MOF
GEF	Graphical Editing Framework
INCITS	International Committee for Information Technology
MDA	Model Driven Architecture
MOF	Meta-Object Facility
OASIS	Organization for the Advancement of Structured Information Standards
OCL	Object Constraint Language
OMG	Object Management Group
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIM	Platform independent model
PIP	Policy Information Point
PPS	Permission <PolicySet>
PSM	Platform Specific Model
RPS	Role <PolicySet>
RBAC XACML Profile	RBAC profile of the XACML 2.0 standard
SSD	Static Separation of Duty
UML	Unified Modeling Language
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
VBAC	View-based Access Control

Chapter 1 Introduction

This thesis describes and illustrates a tool-supported framework that uses the Model Driven Architecture (MDA) approach with the Unified Modeling Language (UML) profile to build the Role Based Access Control (RBAC) application and specification with graphic models and UML notations for distributed system and automatically generates system security specification in eXtensible Access Control Markup Language (XACML) format. This methodology would enable software developers to become more productive and maintain high standards of software quality.

1.1 Motivation

Access control is a security technology that is applied extensively to preventing unauthorized access to information and system resources. Many models have been developed and studied to construct and manage access control systems [1] [2] [3] [4].

Among these models, the Role Based Access Control (RBAC) Model [3] [4] has been widely accepted because of its ability to reduce the complexity and cost for the security administration in large network applications. RBAC regulates the access of users to information and system resources based on the roles granted for users. A role is a job function within an organization with some associated permissions to protected resources. The role-permission assignments and user-to-role authorization rules are defined in policies. Many types of specifications of polices have been proposed [5] [6] [7] [8] [9]. In February 2005, the Organization for the Advancement of Structured Information Standards (OASIS) approved the RBAC profile of the XACML 2.0 standard (RBAC XACML Profile)[10]. This “defines a profile for the use of the OASIS eXtensible Access Control Markup Language (XACML) to meet the requirements for role based access control (RBAC) as specified in the [4]”, where XACML is an XML-based specification for access control that has been standardized in OASIS.

Currently, in most companies, the specification policies for RBAC systems are manually created and managed by a security administrator as an independent procedure during the deployment stage after the software design and development. Because XACML policies are rich XML-based documents with complex syntax, it is very difficult and time consuming to write error free XML documents by hand, especially for people who do not know the syntax well. This may affect the productivity of the administrator and the quality of the software. Hence, a suitable XACML editor or a tool that can automatically generate XACML policies is needed. Another problem is normally the administrator does not take part directly in the application design process, so he/she may not understand the software structure and details well enough to define policies to fulfill the security requirements. This may lead to security risks for the system. This problem can be solved by introducing the RBAC policy design at the beginning of the software design phase and carrying it through the entire development process.

Therefore, this thesis is trying to solve two problems: providing a framework and supported tool to develop the RBAC XACML policy specification along the software design process, and supporting automatic RBAC XACML policy generation.

1.2 Suggested Approach

We propose to use the MDA approach to solve the issues raised in session 1.1. We define a platform independent model (PIM) based on RBAC profile of XACML v2.0 standard and design a UML profile for this metamodel which enables to specify, analyze, design, construct, visualize and document the RBAC system in XACML architecture. We also define the transformation rules to refine PIM models that are prepared within the defined UML profile to Platform Specific Models (PSM) (using the Sun's XACML Implementation as an example). Finally, policy files are generated in XACML format.

1.3 Contributions of this thesis

The major contributions of this thesis are as follows:

- Use the MDA approach to integrate the policy administration with application design process.
- Introducing how to use Object Constraint Language (OCL) to express functions which are defined in the eXtensible Access Control Markup Language (XACML) Version 2.0 [11].
- Extending the RBAC XACML Profile to support the Static Separation of Duty (SSD) RBAC constraints with Role Assignment Policies.
- Designing and implementing a prototype modeling tool RBAC XACML Modeler with the following functions:
    a) Design RBAC modeling and specification as graphic models with UML notation.

b) RBAC policy specification validation

c) Automatic generation of RBAC XACML policies.

1.4 Thesis Outline

This thesis is organized as follows:

Chapter 2 reviews the background of the UML and MDA approach, describes OCL, and introduces tools and software packages used to implement the CASE tool to develop RBAC policies. It also compares this proposed approach with related works.

Chapter 3 describes RBAC models, XACML and the OASIS RBAC profile XACML 2.0 in detail.

Chapter 4 defines the RBAC XACML MDA approach. It defines the RBAC XACML platform independent metamodel, and presents the transformation algorithms from RBAC XACML platform independent metamodel to RBAC XACML policy specifications.

Chapter 5 introduces the CASE tool, the RBAC XACML modeler, which is designed and implemented, based on the MDA approach presented in this thesis. A case study is provided to demonstrate the functionalities of the RBAC XACML modeler

Chapter 6 presents conclusions and future work.

Chapter 2 Background

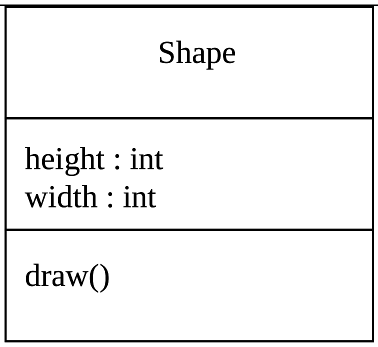
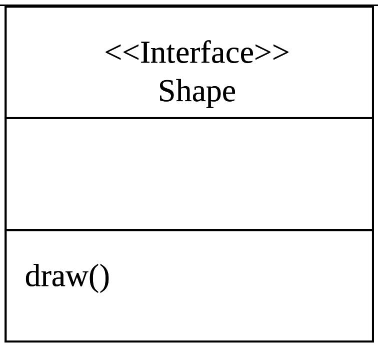
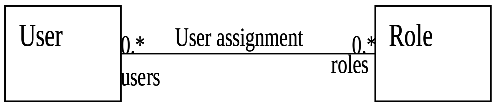
This chapter gives general background information about the UML, the OCL and the MDA approach. It also introduces the Eclipse plug-in, Eclipse Modeling Framework (EMF) and Graphical Editing Framework (GEF) tools that are used to implement the CASE tool in this thesis. Furthermore, the proposed approach is compared with related work.

2.1 UML

The Unified Modeling Language (UML) [12] is a standard modeling language in the field of software engineering. It is a graphic modeling language to help users to specify, visualize, construct and document the components of software systems during the software design and development process. UML consists of a number of diagram types to define the functional, static and dynamic models of a system. The class diagram is one of these defined diagram types to provide a structural view of information in a system.

2.1.1 Class Diagram

Class diagrams are used to show the classes of a system including the attributes and the operations of the classes, and the interrelationships between the classes. A class diagram normally contains the following nodes and edges that are listed in Table 1.

Nodes and edges	Graphic notation	Description
Class		It describes a set of objects that share the same specification of features, constraints and semantics. A graphic notation of Class is a rectangle with three sections, the top one indicates the name of the class, the middle one lists the attributes of the class, and the third one lists the methods.
Interface		Interfaces are classes that have nothing but pure virtual functions. A graphic notation of interface is same as Class plus a stereotype <<Interface>> at the top of the interface name.
Association	 User 0.* users User assignment 0.* roles Role	It is used to describe that a class is related to another class. In the example the two 0...*s indicate that a user object can have 0 or more roles, and a role object can have 0 or more users. These two labels are used to indicate the multiplicity of an

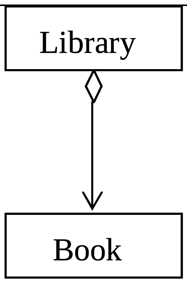
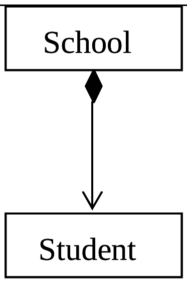
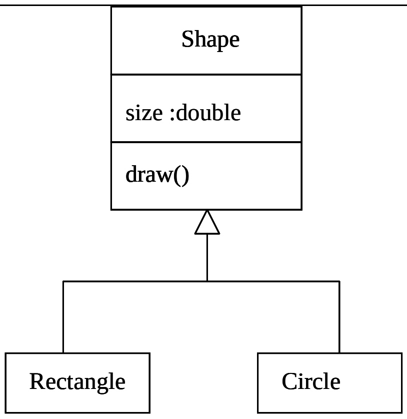
		association.
Aggregation		It describes the whole/part relationship between two classes. In the example, Book is part of Library. Library is a collection of Books.
Composition		Composition is a variation of the aggregation relationship. It is a tightly bound aggregation relationship between two classes.
Generalization		The generalization relationship describes 'is a' or 'is a kind of' relationship. In the example The Rectangle class and the Circle class are children of the Shape class, they inherit the size attribute and the draw () function from the Shape class.

Table 1: Nodes and Edges in UML Class Diagram

2.1.2 OCL

According to UML OCL 2.0 Specification [13] “A UML diagram, such as a class diagram, is typically not refined enough to provide all the relevant aspects of a specification.” Therefore, OCL has been developed as a part of the UML standard to describe additional constraints on the UML models in a formal way [14]. OCL is a textual language that describes constraints on the UML model with expressions. There are four types of constraints defined in OCL, shown Table 2.

OCL constraints type	**Descriptions**
Invariant	An invariant object normally attaches with a class diagram. It is a constraint that states additional rules that must always be obeyed by all objects of the class, type or interface that are defined in the class diagram.
Precondition	A precondition is used to restrict a condition that must be true before an operation executes.
post-condition	A post-condition is used to restrict a condition that must be true after an operation executes.
Guard	A guard is used to restrict a condition that must be true before a transition in a state machine happens.

Table 2 : OCL Constraints Types

The constraints are described by OCL expressions. An OCL expression starts with the keyword “context” and the name of the context to specify the model entity to which the OCL expression is applied, followed by a label to indicate the type of constraint, such as the label inv: declares the constraint type to be an invariant constraint.

For example, suppose we have a simplified UML model for apartment rentals. The following rules are defined for the model:

a) The tenants must be at least 18 years old.

b) The total number of tenants for an apartment must be less than or equal to the maximum occupancy for the apartment.

c) The rental rate of the apartment must be less than 50% of the sum of all tenants' monthly incomes for an apartment.

The model is shown in Figure 1.

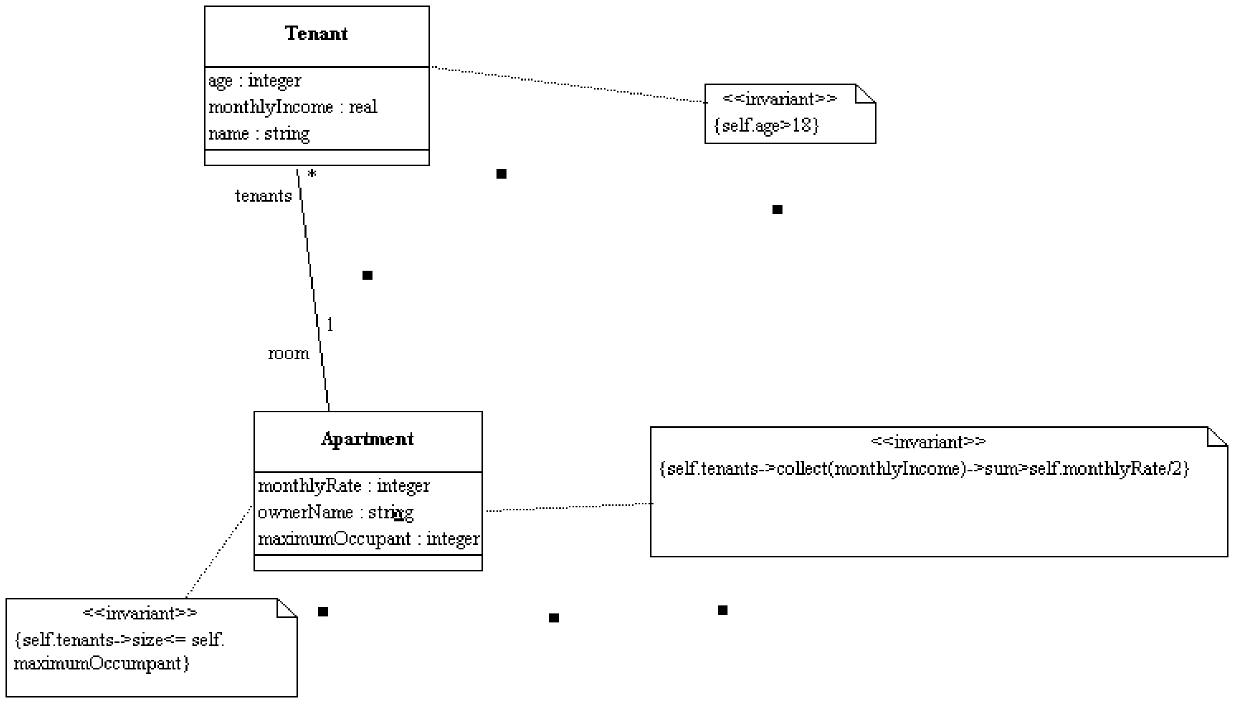


Figure 1: UML Class Diagram with OCL Constraints Example

The OCL rules in the figure are:

a) **Context** Tenant **inv**:

self.age >= 18

This expression indicates that the tenants must be 18 years or older. The "self" is a keyword in the OCL to refer the contextual instance. In this example, it refers to an instance of Tenant. The "age" and "18" are integers. OCL defines a set of basic types

[13]: Integer, Real, String and Boolean. The “>=” operation is one of the standard operations [13] for Real and Integer types defined in OCL.

b) **Context** Apartment **inv**:

self.tenants-> size () <= self.maximumOccupant

This expression indicates that total number of tenants for an apartment must be less than or equal to the maximum number of occupants for the apartment.

The “self.tenants” is a set of Tenant instances. By default, navigation will result in a Set [13]. OCL defines a set of Collection types [13] such as Collection, Set, Bag, OrderedSet, Sequence and also some predefined operations for the Collection types. The navigation notation is “.”. A property of a Collection type is accessed by using an arrow “->” followed by the name of the property. In this example, the size property is used to get the size of the tenants set.

c) **Context** Apartment **inv**:

self.tenants->collect (monthlyIncome) ->sum ()>self.monthlyRate/2

This expression indicates that the rental rate of the apartment must less than 50% of the sum of all tenants’ monthly incomes for an apartment.

The “collect (monthlyIncome)” uses the collect operation, which is one of the predefined OCL collection operations, to specify the collection of monthlyIncome for all tenants in the context of an apartment. The sum operation, which is one of the predefined OCL collection operations, is used to get the additions of elements of the “collect (monthlyIncome)” collection.

2.2 MDA

MDA is an approach to separate the specification of the operation of a system from the implementation details of the system functionality for a particular platform [15]. In MDA, the business model for a particular system domain is abstracted as a platform independent model (PIM), and the system for a particular platform is defined as a platform specific model (PSM) which is derived from the PIM by applying platform specific translation rules.

Figure 2: Model Transformation [15]

In MDA, everything is considered as a model or a model element; hence, it is important to use a well-defined modeling language, such as UML, to describe each model precisely. However, the existing standard modeling languages are so general as to limit their suitability to model a specific domain [14]. To solve this issue, MDA allows the user to define domain specific languages that can precisely formalize specific areas such as business domains, system aspects or particular technologies. Then users can describe PIMs and PSMs based on the defined domain specific metamodels.

There are two alternative approaches to define a domain specific language. The first approach is use the Meta-Object Facility (MOF) [16] to define a new modeling language. The second approach is to extend the UML language with domain specific information, as we will see in Section 2.2.1. The first approach results a new set of semantics to address the domain specific characteristics that are more concise than the UML approach. The new language may use different notations and semantics from UML. Hence, it will not compatible with commercial UML tools. Moreover, users need to learn these new notations and semantics to master this new language. In contrast, these inconveniences will not happen when using the second approach. In our work, we use the second approach.

2.2.1 UML Profile

A UML Profile provides formal semantics to model extensions to the UML metamodel for defining a domain specific language, such as the UML Profile for CORBA [17] which is a standard UML profile defined by Object Management Group (OMG).

In MDA, UML profile plays a very important role in expressing the PIM models, the PSM models, and the transformation rules. UML profiles can be used as a semantic profile, in which they are used to enable the model to accurately and appropriately capture and express specific information. UML profiles are also used as a marking mechanism for tagging a model with extra information needed for model transformation and code generation.

A UML profile uses the stereotype, tagged value and constraints to define the UML extensions. A stereotype defines how an existing metaclass may be extended, and enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass. A stereotype uses the same notation as a Class, with the addition that the keyword «stereotype» shown before or above the name of the Class, a graphic icon or a specific color may be associated with a stereotype as well. As an example, a class with a stereo type Role is shown in Figure 3.

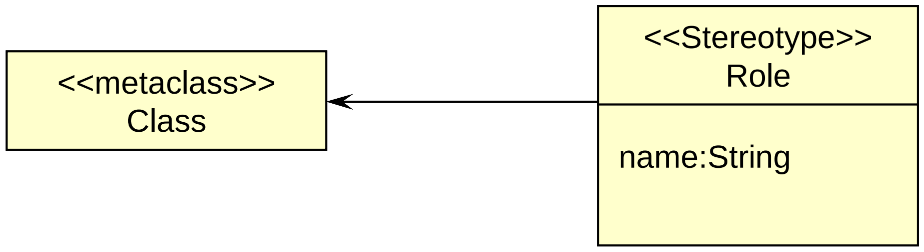


Figure 3: UML Profile Example

When a stereotype is applied to a model element, the values of the properties may be referred to as tagged values. The profile constraints are used to specify the domain restrictions.

2.3 Eclipse Platform

The Eclipse platform is an open source project focused on providing a vendor-neutral open development platform and application frameworks for building software [18]. The Eclipse platform contains a Java Development Environment and a set of seamlessly integrated tools for building software. These tools are built based on plug-ins. A plug-in is the smallest unit of Eclipse Platform function that can be developed and delivered separately. Except for a small kernel known as the Platform Runtime, everything in the Eclipse platform is a plug-in. Anyone can built his/her tools based on existing tools or plug-ins included in the Eclipse platform.

2.3.1 Eclipse EMF and GEF tools

The Eclipse Modeling Framework (EMF) is a modeling framework and data integration tool using the MDA approach to model and generate code for users. EMF allows users to define their PIM models with UML class diagrams, XML documents, or annotated Java, in Ecore model language; the latter is an enhanced MOF2.0 metamodel. EMF also provides tools to generate Java classes from the user defined PIM models, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. EMF can be used to develop other tools and applications based on the EMF structural data model [19] [20].

The Eclipse Graphical Editing Framework (GEF) provides a facility for users to develop graphical editors for an existing model. It can be used with EMF to create graphical editors for EMF based models [19] [21] .

2.3.2 Ecore

Ecore is the meta-meta model used to construct the object model in the EMF framework. It is very similar to the Essential MOF (EMOF) model, a subset of MOF2.0. Ecore and EMOF have roughly same syntax. Hence, EMF can work with, and support standard EMOF XMI model definitions as well. The Ecore components are related according to Figure 4.

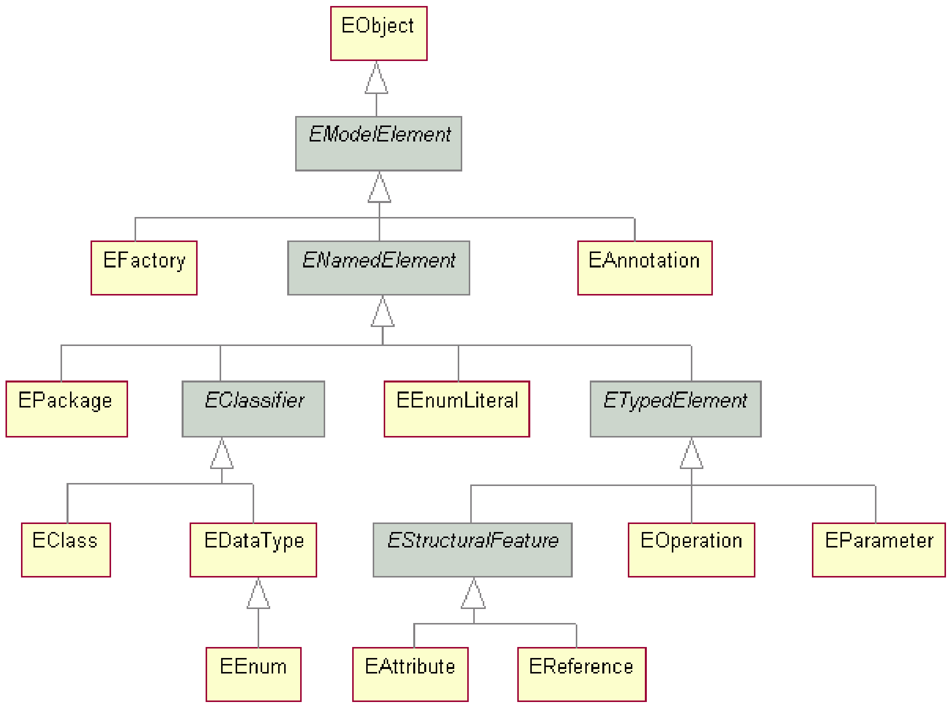


Figure 4: Ecore components[22]

An Ecore model contains classes (and their attributes, references and operations), data types, enumerations, packages and factories. Some of the commonly used Ecore components are explained below [19] [23]:

- EPackage is used same as in Java to collect EClasses and EDataTypes together.
- EClass is used to represent a modeled class. It has a name, zero or more attributes, zero or more references and zero or more operations.
- EAttribute is used to represent a modeled attribute. Attributes have a name and a type.
- EReference is used to represent one end of an association between classes. It has a name, a Boolean flag to indicate if it represents containment, and a reference (target) type, which is another class.

- EDataType is used to represent the type of an attribute. A data type can be a primitive type such as integer or float, or an object type that can be either a library class like java.util.Date or a class defined by the user.
- EOperation is used to represent a modeled operation. Operations have a name, zero or more input parameters and zero or more exceptions.

Our work produces a CASE tool as an Eclipse plug-in and transforms part of the RBAC model into the EMF model.

2.4 Related Works

- UMU-XACML-Editor [24] is a GUI based XACML policy editor software developed by The University of Murcia (UMU). It uses the Java technology to allow user to define XACML 2.0 standard policies. It is unlike our proposed MDA approach that users can model the XACML policies with UML notations. With UMU-XACML-Editor, the users need to define the XACML policies using a form-based graphical user interface (GUI).

- SecureUML [25] and Torsten's MDA VBAC approach [26] are the two most closely related works that use the MDA approach to develop access control systems. Both of these methods allow users to define access control models using the UML notation and use OCL to specify constraints, as in our work. SecureUML is based on general RBAC model and Torsten proposal is based on View-based Access Control (VBAC), which is not used as widely as the general RBAC model. Both of these two works generate access control infrastructures as EJB applications. There is no standard in an EJB application as to how to specify the access control policies. However, our work focuses on providing support for modeling the RBAC systems in XACML architecture and automatic generation of policy specifications in XACML format, which is recognized as a standard format for access control policy specification.

2.5 Chapter summary

This chapter covers the background information that will be used throughout this thesis. Section 2.1 describes the UML modeling language and the OCL constraints that are used with UML to describe a model precisely. Section 2.2 introduces the MDA approach. We explore the components and processes in the MDA and two ways to define the domain specific languages to describe the MDA models. The UML profile is emphasized because it is a lightweight approach to create a domain specific language, and because it is applied in this thesis. Section 2.3 explains the Eclipse platform, Ecore model, EMF and GEF tools, which are used to produce the CASE tool, and part of the RBAC model has been transformed to an Ecore model in this thesis. Section 2.4 lists the related work. The main difference between our work and the related work is that we are focusing on using XACML components to define the RBAC models.

In the next chapter, we are going to introduce more details about the RBAC model and the XACML language.

Chapter 3 RBAC model, XACML and RBAC XACML Profile

This chapter explains the RBAC model and the RBAC XACML profile in detail.

3.1 RBAC model

Using roles to separate access control privileges (Role-Based Access Control) was first introduced in 1992 [3]. An advanced access control framework, "Role Based Access Control Model," was introduced in 1996 [27]. The RBAC model has been widely discussed and further developed since then. In 2001, NIST proposed a consensus model for RBAC, based on these two models. The model was further refined within the RBAC community and has been adopted by the American National Standards Institute(ANSI), International Committee for Information Technology Standards (ANSI/INCITS) as ANSI INCITS 359-2004 [4].

Our thesis uses the RBAC model defined in the ANSI INCITS 359-2004 standard. In this RBAC standard, the NIST RBAC model is defined in terms of four model components:

- CoreRBAC

  The Core RBAC model describes the basic concepts for the RBAC. It is shown in

  Figure 5 .

Figure 5: Core RBAC [4]

These basic sets are:

1. Users(U)

   A user is a human being or an autonomous agent.

2. Roles (R)

   A role is a job function within an organization with some associated permissions to protected resources.

3. Permissions(P) (includes Objects (Obj) and Operations (OP))

   A permission is an authorized right to access or perform operations on protected resources. Objects are the protected resources or tasks in the system. An operation is an executed action performed by a user.

4. Sessions (S)

   When a user logs in the system, he or she establishes a session during which the user activates some roles that are authorized to him or her.

- Hierarchical RBAC

  Normally roles in an organization have overlaps in permission assignments. For example, a professor can have the same permissions as students to read a course mark report, and has additional permissions to add, delete or modify the mark report. Therefore, role hierarchies are introduced as a key aspect of hierarchical RBAC models, in which senior roles acquire the permissions of their juniors. Both general and limited role hierarchies are defined. In general, role hierarchies, roles can have multiple permission inheritance relations. In limited role hierarchies, a role may have one or more immediate ascendants, but is restricted to a single immediate descendant. It is shown in Figure 6.

Figure 6: Hierarchical RBAC [4]

- Static Separation of Duty (SSD) relations

  The RBAC model uses the SSD constraints to specify that two mutually exclusive roles cannot be simultaneously assigned to the same user. The model is shown in Figure 7.

Figure 7: SSD with Hierarchical RBAC [4]

- Dynamic Separation of Duty (DSD) Relations.

  The DSD constraints allow a user to be assigned for two or more mutually exclusive roles that do not create conflicts of interest if the user only acts as one role at any point during a session. However, two mutually exclusive roles or permissions in DSD constraints cannot be activated simultaneously in the same session by the same user. The DSD model is shown in Figure 8.

Figure 8: Dynamic Separation of Duty Relations [4]

3.2 eXtensible Access Control Markup Language (XACML)

XACML is an OASIS standard that defines a general policy language used to protect resources as well as an access decision language based on XML schema. The XACML policy language allows administrators to define the access control requirements for protected application resources. The XACML access decision language is used to represent a query to ask whether a given action should be allowed for a resource. If a policy is found for the queried resource, attributes in the request are compared against attributes contained in the policy rules. Finally a response is determined including an answer about whether the request should be allowed using one of four values: Permit, Deny, Indeterminate (an error occurred or some required value was missing, so a decision cannot be made) or Not Applicable (the request cannot be answered by this service).

3.2.1 XACML architecture

A typical XACML architecture is shown in Figure 9.

Figure 9: XACML Architecture

(1) A user sends a request to a Policy Enforcement Point (PEP).

(2) The PEP forms a request with the subject, resource and action attributes and sends this request to a Policy Decision Point (PDP).

(3) The PDP looks at the request and finds some policy or policy sets that apply to the request from a policy store.

(4) (5) (6) The PDP queries the Policy Information Point (PIP) service to retrieve the attribute values related to the subject, the resource, or the environment and gets a response with desired attribute values.

(7) The PDP compares attributes in the request against attributes contained in the policy rules and comes up with an answer about whether access should be granted. That answer is returned to the PEP, which can then allow or deny access to the requester.

3.2.2 XACML Policy Language

3.2.2.1 Policy language components

As shown in Figure 9, request/response, Policy, and PolicySet are defined in the XACML policy language. The XACML policy language components are shown in Figure 10.

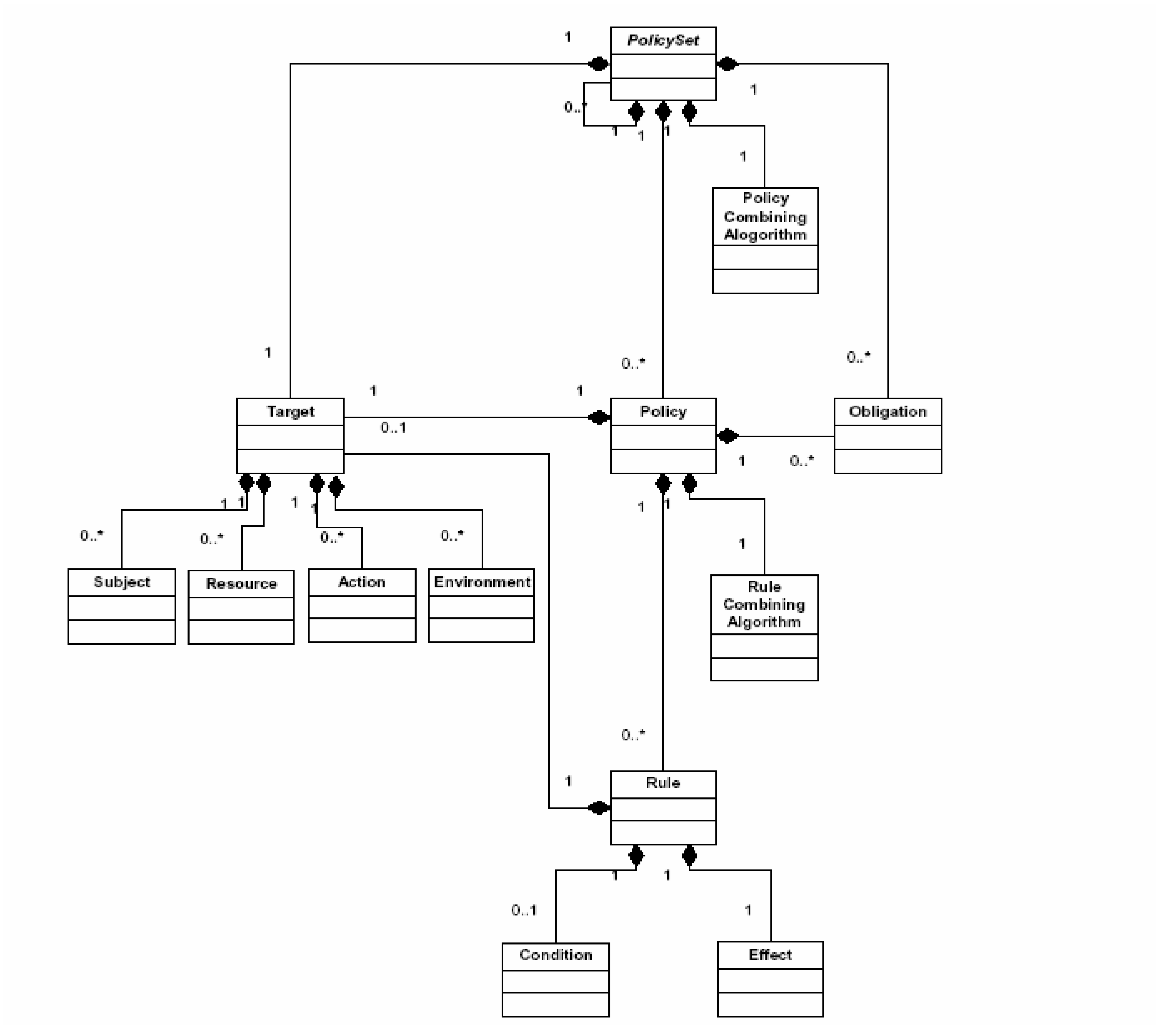


Figure 10: XACML Policy Language Model [11]

- PolicySet contains the following subcomponents:

  a) A set of Policies, or PolicySets as well as references to remote policies.

  b) A Target to which this PolicySet is intended to apply

  c) An Obligation, which is the actions that must be performed by the PEP in conjunction with the enforcement of an authorization decision.

  d) A Policy combining algorithm, which is used to reconcile multiple results into a single decision.

Note: Since a PolicySet may contain multiple Policies that apply to the request and each policy may result in a different access decision, the Policy Combining Algorithm is required to reconcile multiple results into a single decision. Examples of Policy Combining Algorithms include policy deny overrides (ordered and unordered), policy permit overrides (ordered and unordered), policy first applicable, and policy only-one-applicable.

- Policy contains the following subcomponents:
    a) A set of rules
    b) A target to which this policy is intended to apply
    c) An obligation, which is the actions that must be performed by the PEP in conjunction with the enforcement of an authorization decision.
    d) A rule-combining algorithm, which is used to reconcile multiple results into a single decision.
- Rule contains following subcomponents:
    a) A target to which this rule is intended to apply
    b) A condition that is a Boolean expression used to evaluate attributes.
    c) An effect which is an access decision defined by policy writer. Its value only can be "Permit" or "Deny".

3.2.2.2 An Policy Example

Table 3 (next page) demonstrates the access control policy that states anybody is permitted to perform a read action on the protected file "Student List".

```
1       <Policy PolicyId="policy1"   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
2       combining-lgorithm:permit-overrides">
3       <Target>
4         <Subjects>
5               <AnySubject/>
6         </Subjects>
7         <Resources>
8               <AnyResource/>
9         </Resources>
10        <Actions>
11              <AnyAction/>
12        </Actions>
13      </Target>
14      <Rule RuleId="Rule1" Effect="Permit">
15        <Target>
16          <Subjects>
17              <AnySubject/>
18          </Subjects>
19          <Resources>
20              <Resource>
21                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
22                <AttributeValue
23                DataType="http://www.w3.org/2001/XMLSchema#string">StudentList</AttributeValue>
24                <ResourceAttributeDesignator
25                               AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
26                               DataType="http://www.w3.org/2001/XMLSchema#string"/>
27                </ResourceMatch>
28              </Resource>
29          </Resources>
30          <Actions>
31              <Action>
32                <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
33                <AttributeValue
34                DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
35                <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
36                DataType="http://www.w3.org/2001/XMLSchema#string"/>
37                </ActionMatch>
38              </Action>
39          </Actions>
40        </Target>
41      </Rule>
42        </Policy>
```

Table 3: XACML Policy Example

Line 1 and Line 2 defines a name to the policy instance and specifies the algorithm that will be used to resolve the results of the various rules that may be in the policy. In this example, “permit-overrides” algorithm is used; therefore, if any rule evaluates to “permit”, then the policy returns “Permit”.

Lines 3 through 13 define the decision requests to which subject, resource, and action this policy applies. In this example, the target section says the policy is applicable to any decision request.

Line 14 defines a rule in this example and its effect is “Permit,” if this rule is evaluates to “True”.

Lines 15 through 40 define the decision requests to which subject, resource, and action this rule applies. Lines 15 through 18 describe the rule that applies the decision requests to any subject. Lines 19 through 29 describe the rule that applies the decision requests to a specific resource value, which must match the <ResourceMatch> element. The <ResourceMatch> element specifies a matching function in the MatchId attribute, a string value “StudentList,” and a pointer to a specific “resource attribute” in the request context, by means of the <ResourceAttributeDesignator> element. The match function is used to compare the string value with value of the “resource attribute” in the request context. This rule applies to a decision request, only if the match returns “True”. Lines 30 through 39 define the action match in a similar format to a resource match. It defines the match of “action attribute” in the request context to string value “read”.

Line 41 closes the Rule element.

Line 42 closes the Policy element.

3.3 RBAC profile of XACML v2.0 standard

This standard “defines a profile for the use of the OASIS Extensible Access Control Markup Language (XACML) to meet the requirements for role based access control (RBAC) as specified in [the Proposed ANSI Standard, BSR INCITS 359]”.

3.3.1 Components

The RBAC profile of XACML v2.0 standard specifies four types of policies to build a RBAC solution [10].

- Permission <PolicySet> or PPS :

  As shown in Figure 11, PPS is a collection of permissions associated with a role. It is a <PolicySet > that contains a set of polices which defines permissions associated with the given role directly, and a set of PPS references which associate with other roles that are junior to the given role. In order to support the hierarchy model, it required that the <Target> element of a PPS if present, must not limit the subjects to which the <PolicySet> is applicable.

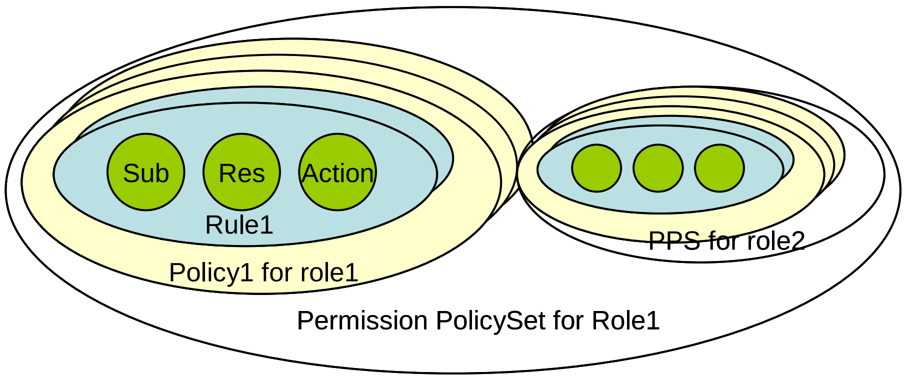


Figure 11: Permission PolicySet (PPS)

- Role <PolicySet> or RPS

  As shown in Figure 12, an RPS connects a role to the corresponding PPS that contains actual permissions associated to the given role. Each RPS contains a <Target> element indicating the applicable role and a reference to PPS.

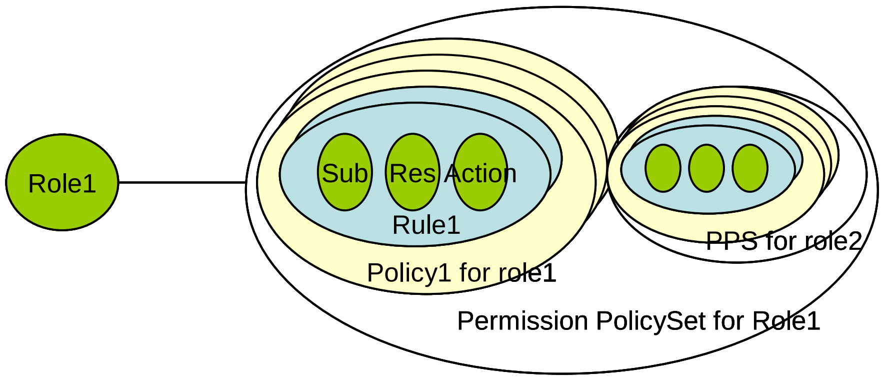
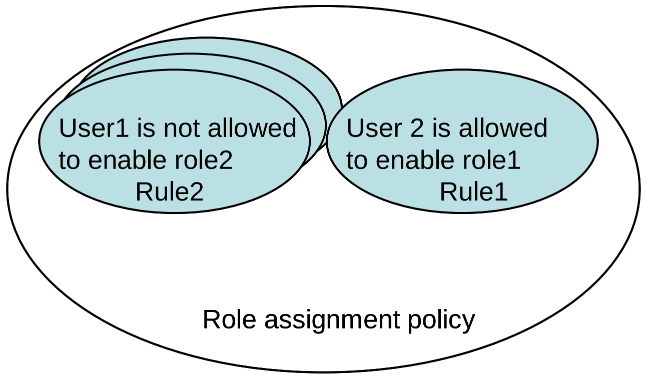


Figure 12: Role PolicySet (RPS)

- Role Assignment <Policy> or <PolicySet>

  This type of policy is optional. It is used to answer the question as to whether a subject is allowed to be assigned to a role for the user-to-role authority. It is shown in Figure 13.



User1 is not allowed to enable role2
Rule2
User 2 is allowed to enable role1
Rule1
Role assignment policy

Figure 13: Role Assignment Policy

- HasPrivilegeOfRole <Policy>

HasPrivilegeOfRole <Policy> is used to answer the question as to whether a subject has the privileges associated with a given role. This type of policy is optional.

3.4 Chapter Summary

This chapter introduces the details about the RBAC models and XACML languages.

Session 3.1 explains the four model components defined in the RBAC (ANSI INCITS 359-2004) standard. Session 3.2 explains the basic components in an XACML architecture system and the policy language components. Session 3.3 explores the RBAC Profile for XACML, which defines how to use XACML language to describe a RBAC model. Based on the above foundation and the background information introduced in chapter 2, we are ready to define the PIM model and the transformation rules used in this thesis.

Chapter 4 The RBAC XACML – MDA approach

4.1 Overview

This section gives an overview of our model driven development process for developing a RBAC XACML policy specification model.

Figure 14 : RBAC Model Driven Development Process

As shown in Figure 14, at first, the software developer designs the RBAC XACML specification platform-independent models (RBAC XACML PIM) based on the access control requirements along with the application models . Then, the designed RBAC XACML PIM models are transformed into the RBAC XACML specification platform-specific models (RBAC XACML PSM) which are used to generate security infrastructure files. In our research work, we use the SUN XACML implementation as the RBAC PSM model and the Eclipse EMF model for application PSM models. These PSM models can be further used to generate RBAC XACML policy specifications and application code. After automatically generating the system security infrastructure, the software developers implement missed parts based on generated skeletons, build the system, and test the system.

We will give more details about the RBAC XACML PIM model and the model transformation in following sections.

4.2 RBAC XACML-PIM Meta Model

The RBAC XACML-PIM metamodel shown in Figure 15, is defined based primarily upon the OASIS RBAC profile of the XACML 2.0 standard and an extension to support SSD constraints. The primary elements are Role, RPS, PPS, Policy, Rule and Resource. The role to resource permission assignment is a composite of a RPS (Role Policy Set) component and a PPS (Permission Policy Set) component where the PPS component defines actual permissions for a role and the RPS component associated this role to the corresponding PPS component. The PPS component includes a set of permission policies and references to its junior role's PPS. The policy component includes a set of rules that defines actions on the protected resources. User to role authority assignment is defined in the Role Assignment component. The checking for exclusive roles enablement is added into role assignment policy set as applying conditions to support the SSD constraints.

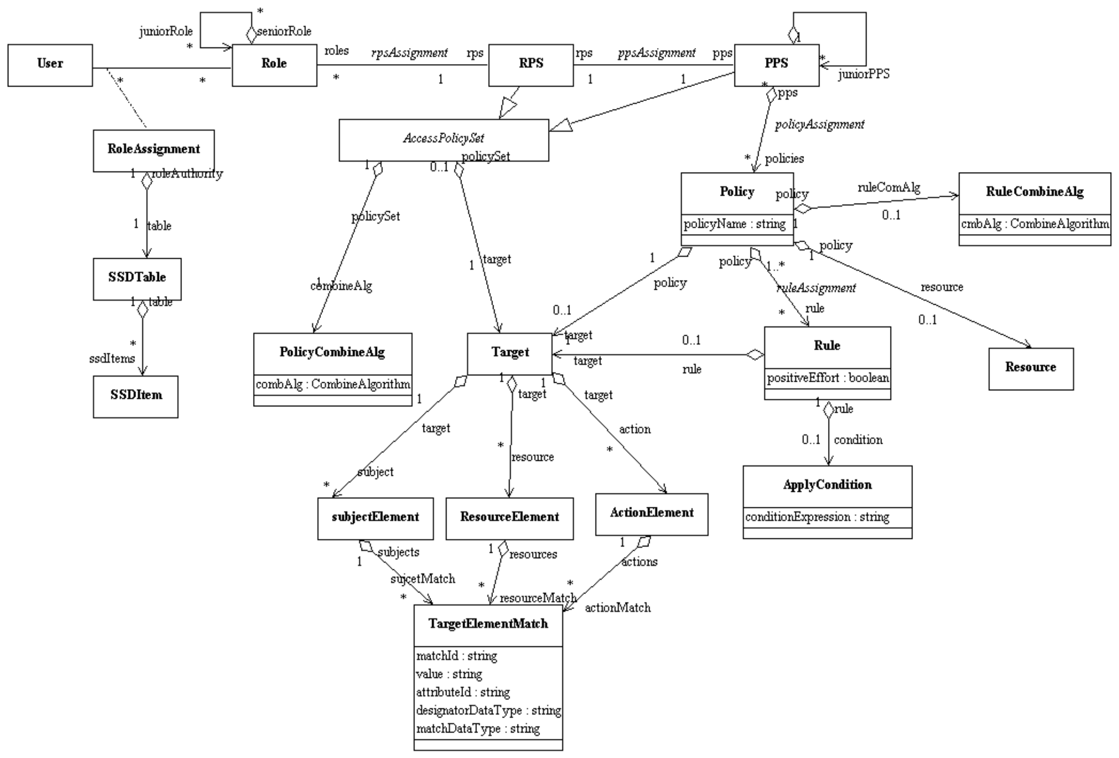


Figure 15: RBAC XACML PIM Metamodel

4.2.1 User

A user represents a person or a subject in the system. A user may be assigned to several roles, but that user cannot play two mutually exclusive roles that are defined in an SSD (4.2.3) at the same time.

4.2.2 Role Assignment

The Role assignment component is an association class. It is used to compose the role assignment policy set, which defines user-role assignments with conditions that no two mutually exclusive roles are assigned to a user at the same time. When a role assignment association is created, it checks against the SSD table to avoid the user associating to two mutually exclusive roles simultaneously.

4.2.3 SSDTable

SSDTable is used to store all mutually exclusive roles. It contains a collection type attribute:

- ssdItems[*] : String

  The ssdItem in ssdItems collection is a String with "{role1, role2}" format, where role1 and role2 represent two roles which are mutually exclusive.

4.2.4 Role

A role represents a job function that has privileges to execute the actions on protected resources. It contains an RPS (4.2.13) for the role-permission assignment and an aggregate association for the role hierarchy. A senior role gets all the permissions that its junior roles have.

4.2.5 AccessPolicySet

An AccessPolicySet component defines an abstract class that contains a target element (4.2.8) which is used to identify an applicable policy set for the decision request, and a policyCmbAlg attribute (4.2.7) which is used to reconcile the individual evaluation result of a set of policies into a single decision. Its subclass contains a set of policies or policy sets.

4.2.6 CombineAlgorithm

CombineAlgorithm is an enumeration type. It specifies the procedure by which multiple algorithm evaluation results are combined and a final decision is computed.

The available policy combining algorithms are as following:

- deny-overrides

If a single evaluation result is “Deny”, then regardless of other evaluation results, the combined result is Deny.

- permit-overrides

  If a single evaluation result is “Permit”, then regardless of other evaluation results, the combined result is Permit.

- first-applicable

  The combined result is the same as the first evaluation result.

- only-oneapplicable

  This combining algorithm is only available for PolicyCombinAlg (4.2.7), which is used to specify the combining algorithm for a set of policies. The algorithm is used to ensure only one policy or policy set is applicable to the decision request. The combined result equals “NotApplicable” for no applicable policy or policy set, “Indeterminate " for more than one applicable policy or policy set, or the evaluation result for the exactly one applicable policy.

- ordered-denyoverrides

  This combining algorithm is similar to deny-overrides except that the evaluation is processed in a defined order.

- ordered-permitoverrides

  This combining algorithm is similar to permit-overrides except that the evaluation is processed in a defined order.

4.2.7 PolicyCombiningAlg

PolicyCombiningAlg represents the combine algorithm (4.2.6) for a set of policies or policySets. It contains an Enumeration Type attribute combAlg.

4.2.8 Target

A Target contains a subject (4.2.10) element, a resource (4.2.19) element and an action (4.2.12) element. It specifies an object to which a policy set, a policy or a rule applies. If it is set to null, all policy set, policies or rules are applicable.

4.2.9 TargetElement

TargetElement defines the common data structure for matching targets.

It contains the following attributes:

- matchId : String

  Specifies the name of a XACML predefined match function.

- attributeDataType : String

  Specifies a XACML predefined data type for the matching attribute.

- value : String

  Defines the value of the matching attribute.

- designatorDataType : String

  Specifies the data type of an attribute designator.

- attributeId : String

  Specifies the attribute Id of an attribute designator.

  Note: In XACML, the Attribute Designator element is used to retrieve attribute values from a request by specifying the name, type, and issuer of attributes.

4.2.10 SubjectElement

The SubjectElement component contains a set of subject target (4.2.9) matches. It is used to define the matching conditions for the subject attributes within a decision request. If it is set to null, the target matches all subjects.

4.2.11 ResourceElement

The ResourceElement component contains a set of resource target (4.2.9) matches. It is used to define the matching conditions for resource attributes within a decision request. If it is set to null, the target matches all resources.

4.2.12 ActionElement

The ActionElement component contains a set of action target (4.2.9) matches. It is used to define the matching conditions for action attributes within a decision request. If it is set to null, the target matches all actions.

4.2.13 RPS

The RPS component is a subclass of AccessPolicy (4.2.5). It represents the Role policy set which associates a role or roles (for multi-role permissions) to the actual permissions (PPS – 4.2.14). In its target element, the subjectElement (4.2.10) is defined to match all its associated roles to limit the applicability of the policy set. Normally, an RPS component associates with only one role. Sometimes, it associates with multiple roles to meet the circumstance where some policies need a user to hold several roles at same time to gain the permissions. A RPS component associates with only one PPS (4.2.14) component.

4.2.14 PPS

The PPS component is a subclass of AccessPolicy (4.2.5). It contains a set of policies and references to the PPS components that are associated to its junior roles. In its target element, the subjectElement (4.2.10) is set to null; hence there is no limit to the subjects to which the PPS is applicable.

4.2.15 Policy

The Policy component specifies the actual access rules to a resource. It contains a set of rules, an identifier for the rule combining algorithm (4.2.16), a target (4.2.8) and a resource (4.2.19). In its target element, the subjectElement (4.2.10) is set to null; hence there is no limit to the subjects to which the policy is applicable. The resource element is used to identify protected resource to which this policy applies.

4.2.16 RuleCombiningAlg

The RuleCominingAlg component represents the combine algorithm (4.2.6) for a set of rules. It contains an Enumeration Type attribute combAlg.

4.2.17 Rule

The rule is the basic component in a policy. It has one attribute:

- positionEffort : Boolean

  The effect is “Permit”, if positionEffort equals true.

  The effect is “Deny”, if positionEffort equals false.

Each rule also contains a condition (4.2.18) which is used to evaluate the rule.

4.2.18 ApplyCondition

The ApplyCondition component represents a Boolean expression to evaluate the rule. The outcome of the rule depends on the condition evaluation. If the condition returns False, the rule returns NotApplicable. If the condition returns true, the value of the Effect element is returned, which is either Permit or Deny. It has an attribute:

- conditionExpression : String

  It uses OCL syntax to express the evaluation constraint.

4.2.19 Resource

The resource component represents the protected resource. It is associated with policies. Each resource has attributes and operations.

4.3 UML profile for the RBAC XACML-PIM Meta Model

As introduced in section 2.2, we use the UML profile to represent the RBAC XACML PIM metamodel. The mapping between each element of the PIM metamodel and the stereotypes, constraints and tagged values, are shown in Table 4 (next page).

UML Metamodel type	**Stereotype**	**Tag values**	**RBAC XACML PIM Metamodel Type**
Class	<<User>>		User
Class	<<Role>>		Role
Class	<<Resource>>		Resource
Association	<<RoleAssignment>>		Role Assignment
Class	<<SSDTable>>		SSDTable
Attribute			SSDItem
Generalization			Role Hierarchy
Class	<<RPS>>		RPS
Class	<<PPS>>		PPS
Association	<<Permission Link>>		rpsAssignment ppsAssignment policyAssignment ruleAssignment
Class	<<Permission>>		Policy
Class	<<Rule>>	positiveEffort:Boolean	Rule
Class	<<Apply>>	conditionExpression:String	ApplyCondition
Class	<<SubjectTarget>>	matchId:String value:String attributeId:String designatorDataType:String matchDataType:String	SubjectElement
Class	<<ResourceTarget>>	matchId:String value:String attributeId:String designatorDataType:String matchDataType:String	ResourceElement
Class	<<ActionTarget>>	matchId:String value:String attributeId:String designatorDataType:String matchDataType:String	ActionElement

Table 4: UML Profile for the PIM metamodel

4.4 Generating the RBAC XACML Policy Specifications

This section represents how to generate the RBAC XACML Policy Specifications and the infrastructure responsible for enforcing permissions. Sun's XACML Implementation [28] has been selected as the specific platform. The mapping between RBAC XACML PIM to Sun’s XACML Implementation is shown in Table 5.

RBAC XACML–PIM	**SUN XACML Implementation**
RPS, PPS, RoleAssignment	PolicySet
PolicyCombiningAlg	PolicyCombiningAlgorithm
TargetElement	TargetMatch
Target	Target
Policy	Policy
RuleCombiningAlg	RuleCombiningAlgorithm
Rule	Rule
ApplyCondition	Apply

Table 5 : PIM metamodel mapping to the Sun XACML Implementation

The User component, the Role component and the Resource component are mapped to Ecore model components, and an Ecore model is generated for these components.

The mapping is shown in Table 6.

RBAC XACML–PIM	**Ecore Model**
User Role Resource	EClass
Attributes of User, Role and Resource	EAttribute
Methods of User, Role and Resources	EOperation

Table 6: Mapping between User, Role and Resource components to ECore model

4.4.1 Generating User, Role and Resource Classes

As shown in Table 6, the User component, the Role component and the Resource component in RBAC XACML–PIM are transformed to EClass objects of the EMF model, their attributes map to EAttribute objects, and their methods map to EOperation objects. Hence, an EMF model is created for these three components. Afterward, the regular Java classes with customized attributes and methods for these three components can be produced based on the created EMF model.

4.4.2 Generating policy files

As shown in Table 5, The Policy, RPS, PPS and Role Assignment components are transformed to the Policy Class or the PolicySet class. After the corresponding Policy and PolicySet objects are created, these objects can be encoded into XACML files.

4.4.2.1 Policy

The RBAC XACML PIM Policy component contains a policyName attribute, a Target element, a RuleCominingAlg element and a set of Rule elements. It maps to the Policy Class in Sun XACML implementation. The procedures of transforming a Policy component to a Policy object are described below:

1. Creation of the RuleCombiningAlgorithm for the Policy object:

   The RuleCominingAlg has an enumeration type attribute comAlg. It is transformed to a string variable algRuleId shown in Table 7. The transformed string value is used to create a RuleCombiningAlgorithm object.

RuleCombiningAlg.comAlg	algRuleId:String
deny-overrides	"urn:oasis:names:tc:xacml:1.0:rule-combining-

	algorithm:deny-overrides"
Permit-overrides	"urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides"
firstapplicable	"urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable"
ordered-denyoverrides	"urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:ordered-deny-overrides"
ordered-permitoverrides	"urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:ordered-permit-overrides"

Table 7 : RuleCombiningAlg transformation

A code fragment for creating a RuleCombiningAlgorithm object is shown below:

Table 8 : Creating an RuleCombiningAlgorithm Object

2. Creation of the target for the Policy object.

   As described in RBAC XACML Profile, a Target element of PPS, and its included or referenced PPS, Policy, and Rule elements, shall not limit the subjects to which the PPS is applicable [10]. Therefore, a default target is created with no limitation to subjects, resources and actions. The actual limitations of resources and actions are defined in rule target.

   A code fragment for creating a Target Object for the Policy object is shown below:

Table 9 : Creating a default Target Object

3. Creation of the rules for the Policy object.

   The associated rule elements of the Policy component are transformed to the Rule Class. The procedures for rules transformation are described in section 4.4.2.2.

4. Creation of the policy object.

Table 10 : Creating a Policy Object

4.4.2.2 Rule

The RBAC XACML PIM Rule component contains a positiveEffort attribute, a Target element, and an ApplyCondition element. It maps to the Rule Class as shown in Table 5. The procedures to transform a Rule component to a Rule object are described below:

1. Definition of the effect attribute for the Rule object.

   The positiveEffort attribute in the Rule component maps to the effort attribute of the Rule object. Mapping between the "positiveEffort" and "effort**"** is shown below:

positiveEffort :Boolean	**effort : int**
True	Result.DECISION_PERMIT
False	Result.DECISION_DENY

Table 11 : The Effort transformation

2. Creation of the rule target object for the Rule object.

A Target object in Sun XACML is constructed with a list of subjects, a list of resources and a list of actions, where each list contains a list of TargetMatch objects that defines an attribute and its values. TargetMatch Class is constructed with a target match type, which must be one of the action, resource or subject type, a match function, an AttributeDesignator Object that defines the match attribute, and an AttributeValue object that defines match attribute’s data type and value.

The RBAC XACML PIM Target component maps to the Target Class (Table 5). The Target component contains a list of SubjectElement components, a list of Resource components and a list of ActionElement components. These three lists map to subjects, resources and actions of the Target Class respectively. The SubjectElement, ResourceElement and ActionElement components inherit the TargetElement component that maps to the TargetMatch Class. The TargetElement component contains following attributes:

- matchId :String

  The matchId is used to create the match function, but matchId must be one of the SUN XACML supported function strings.

Table 12 : Creating a XACML predefined function

- designatorDataType :String
- attributeId :String

The designatorDataType and attributeId are used to create an AttributeDesignator object, where the designatorDataType must be one of SUN XACML supported types strings [28]

Table 13 : Creating an AttributeDesignator object

- attributeDataType :String
- value :String

The attributeDataType and value are used to create an AttributeValue object.

Table 14 : Creating an AttributeValue object

Putting all of the above sub-components together, a TargetMatch Object is created:

Table 15 : Creating a TargetMatch object

After three TargetMatch lists are created for the SubjectElement, the ResourceElement, and the ActionElement respectively, a Target object is created as below.

Table 16 : Creating a Target object

3. Creation of the condition for the Rule object

The RBAC XACML PIM ApplyCondition component maps to the Apply Class (Table 5). The Apply class is used to further refine the applicability of the rule that is applicable for the Target requirements. An Apply object is constructed with:

- A comparison function.
- A list of comparison values: includes an AttributeDesignator object to specify which value is to be extracted from the request and an AttributeValue object to specify the expected value.
- An optional Higher Order Bag function: it is used to handle the case of multiple attribute values.

The ApplyCondition component has a string attribute conditionExpression, which uses the OCL syntax to express the condition constraint. The conditionExpression is parsed to create an Apply object. The transformation rules are described below:

- Creation of a comparing function.

  The mapping between OCL predefined data operations to Sun XACML Condition Functions is shown in Table 17 (next page), where “*” represents appropriate data type defines in XACML[11].

OCL operations	**Function Identifier**
a = b	*-equal
a > b	*-greater-than
a>=b	*-greater-than-or-equal
a<b	*-less-than
a<=b	*-less-than-or-equal
a/b	*-divide
a.abs()	*-abs
a+b	*-add
a.floor()	urn:oasis:names:tc:xacml:1.0:function:floor
a.mod()	urn:oasis:names:tc:xacml:1.0:function:integer-mod
a*b	*-multiply
not a	urn:oasis:names:tc:xacml:1.0:function:not
a.round()	urn:oasis:names:tc:xacml:1.0:function:round
a-b	*-subtract
a.includes(b)	*-is-in
a.size()	*-bag-size
a.bag()	*-bag
a.size()=1	*-one-and-only
a.forAll(exp1 forAll(exp2))	urn:oasis:names:tc:xacml:1.0:function:all-of-all
a.forAll(exp)	urn:oasis:names:tc:xacml:1.0:function:all-of
a.exists(exp)	urn:oasis:names:tc:xacml:1.0:function:any-of
a.forAll(exp1 exists(exp2))	urn:oasis:names:tc:xacml:1.0:function:all-of-any
a.exists(exp1 forAll(exp2))	urn:oasis:names:tc:xacml:1.0:function:any-of-all
a.exists(exp1 exists(exp2))	urn:oasis:names:tc:xacml:1.0:function:any-of-any
And	urn:oasis:names:tc:xacml:1.0:function:and
Or	urn:oasis:names:tc:xacml:1.0:function:or

Table 17: Mapping OCL expressions to Sun XACML condition functions

Note: Only a subset of XACML functions have been covered in this table. S some functions such as date or math functions are not included in this table.

The transformed “Function Identifier” is used to create the compare function object. For example, the following code fragment (Table 18) creates a comparison function object for “a=b”, where the data type of “a” and “b” is string.

Table 18 : Creating a Function object

- Creation of an AttributeDesignator object.

The attribute operation in the conditionExpression, such as “source.attribute”, is parsed to create an AttributeDesignator object where the “source” is an instance of Role or Resource component, and the “attribute” is an attribute defined in the “source” class. The mapping rules are defined in Table 19.

source.attribute	**new AttributeDesignator(int target, URI type, URI id, boolean mustBePresent)**
If source is an instance of Role component	target = TargetMatch.SUBJECT
If Source is an instance of Resource component	target = TargetMatch.RESOURCE
attribute’s data type	type = new URI(corresponding XACML standard data type);
attribute’s name	id = new URI (attribute’s name);

Table 19 : Parsing OCL expression example

The AttributeDesignator always returns the values for an attribute in a bag, which is an unordered collection. The returned bag may contain multiple values. However, only one value is needed for the comparing function, hence the bag function “*-one-and-only” is needed to guarantee that the AttributeDesignator returns a bag with exactly one value.

Table 20 : Creating an Apply object

Finally, the variable value in conditionExpression is used to create the AttributeValue.

Putting everything together, an Apply object is created for a conditionExpression string. Table 21 shows the code fragment for creating an Apply object for a conditionExpression string “order.sum<1000”, where order is an instance of a Resource component and sum is an integer attribute.

Table 21 : Creating Apply condition for "order.sum<1000"

4.4.2.3 PPS

A PPS component contains a policySetName attribute, a Target element, a PolicyCombiningAlg element and a set of Policy elements. It maps to the PolicySet Class.

A PolicySet instance is constructed with a policy Id, a PolicyCombiningAlgorithm object, a Target Object and a list of AbstractPolicy objects. The procedures of transforming a PPS component to a PolicySet object are described below:

1. Create a PolicyCombiningAlgorithm object for the PolicySet object:

   The PolicyCombiningAlg contained by the PPS component has an enumeration type attribute comAlg. It is transformed to a string variable algPolicyId as shown in Table 22. The transformed string value is used to create a PolicyCombiningAlgorithm object.

PolicyCombiningAlg.comAlg	algPolicyId:String
deny-overrides	"urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides"
Permit-overrides	"urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides"
firstapplicable	"urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable"
only-oneapplicable	"urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable"
ordered-denyoverrides	"urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:ordered-deny-overrides"
ordered-permitoverrides	"urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:ordered-permit-overrides"

Table 22 : PolicyCombiningAlg transformation

A code fragment for creating a PolicyCombiningAlgorithm object is as below:

Table 23 : Creating a PolicyCombiningAlgorithm object

2. Creation of a Target object for the PolicySet object.

   As explained for the Policy's object creation in section 4.4.2.1, a default Target object is created.

3. Creation of an AbstractPolicy list for the PolicySet object.

   The AbstractPolicy list contains all Policy Objects associated with this PolicySet Object and all PolicyReference objects that hold references to its junior roles' permission policy sets.

   A Policy object is created for every Policy element contained by the PPS component as described in section 4.4.2.1. A PolicyReference is created for every junior role's permission policy set. The Sun XACML implementation supports policy/policySet references, but does not include a PolicyFinderModule that implements policy/policySet references. Users need to implement their own PolicyFinderModule class to find the referenced policy/policySet. Therefore, during the transformation, a PolicyFinderModule class is developed to find the referenced permission policy set.

4. Create a PolicySet object for the PPS component.

Table 24 : Creating a PolicySet object

4.4.2.4 RPS

A RPS component contains a policySetName attribute, a Target element, a PolicyCombiningAlg element, a set of Role elements and a PPS element. It maps to the PolicySet Class. The procedures to transform a RPS component to the PolicySet class are described below:

1 Create the PolicyCombiningAlgorithm object:

This step is the same as creating a PolicyCombiningAlgorithm object for a PPS component, as described in section 4.4.2.3.

2 Create the Target object.

Instead of creating a default Target object as the PPS component 4.4.2.3, the Target object for RPS PolicySet object contains a subject TargetElement which require a given Subject to have role attributes or values matched to the Role elements contained by this RPS component. This Target object has no restriction as to resource and object elements. The code segment to create a Target object is similar to the one shown in section 4.4.2.1.

3 Create the AbstractPolicy list.

The AbstractPolicy list only contains a PolicyReference object, which holds a reference to its PPS PolicySet object. The same PolicyFinderModule class as in section 4.4.2.3 is used to create the PolicyReference object.

4 Create a PolicySet object for the RPS component.

This step is same as described in section 4.4.2.3

4.4.2.5 Role Assignment

A Role Assignment component associates users with roles to enable user-to-role assignments. It also contains a SSD Table component to store the mutually exclusive roles. It maps to the PolicySet Class. In role assignment, PolicySet uses a Subject TargetElement to identify the user who requires a role enablement, a Resource TargetElement to express roles which are allowed to be assigned to a given user, an Action TargetElement to match the "enableRole" action in the decision request, and an Apply condition to check if the user requires to enable a role which is mutually exclusive to its active roles.

4.4.3 Generating basic PDPs and PEPs

As shown Figure 9, the PDP and the PEP are two major components in the XACML architecture. The PEP receives a request for access from a user, formulates and sends the request to the PDP, and interprets the PDP response. The PDP finds the applicable policy or policySet (only one policy or policySet is applicable for a decision request), evaluates the request against the applicable policy or policySet, and sends the evaluation results back to the PEP.

- PDP

In the Sun XACML Implementation, the PDP needs a set of policy finder modules (PolicyFinderModule) to access policies, retrieve attribute values and resolve resources to evaluate the decision request. Therefore, a RolePolicyFinderModule class is developed during the transformation to find, to access and to evaluate the role assignment policySet and the role permission policySets.

- PEP

The PEP creates requests based on the requesters' attributes, the actions which the requesters wish to execute and the resources to which the requester would like to access. Hence, requests need to be created in real time. After a request is created, the PEP sends the request to PDP and interprets the response. In addition, because there is currently no standard way to send XACML requests to an online PDP, users needs define their own way to pass requests and responses between the PEP and the PDP. Because of the above reasons, only a skeleton of the PEP class is implemented during the transformation, and the developers need to complete the implementation manually.

4.5 Chapter summary

This chapter presented the RBAC XACML PIM metamodel, and the procedures of transforming the RBAC XACML PIM metamodel to the Sun XACML implementation model. An XACML architecture and policy specifications are generated during the transformation. An EMF model for the User, Role and Resources components is created for future development as well.

We have defined a PIM model and a UML Profile to describe RBAC models with XACML components. We also presented the mapping details from the defined PIM model to the Sun implemented XACML application, and part of the PIM models has been transformed to EMF models.

The next chapter will introduce a CASE tool that can deploy notations defined in RBAC XACML UML profile to model the RBAC XACML application and generate both the Sun's XACML implementation model and the EMF model.

Chapter 5 Modeling with RBAC XACML Modeler

5.1 Overview

In this chapter, we present a CASE tool, “RBAC XACML Modeler”. It is designed and implemented as a prototype tool to explain how to use the proposed MDA approach to specify, design and construct the RBAC system using UML notation, and automatically generate the XACML policy specifications.

5.2 RBAC XACML Modeler

A UML CASE tool, RBAC XACML Modeler, is developed as an Eclipse plug-in using the EMF and GEF frameworks to support the MDA approach introduced in this thesis.

The RBAC XACML Modeler has following functions:

- Provides a visual editor to allow a user to view and edit RBAC XACML models graphically.
- Automatically transforms and generates policy files in XACML format.
- Creates an EMF model for User, Role and Resource components in RBAC XACML models.
- Automatically validates the model to avoid an invalid design.

A screen shot of the RBAC XACML Modeler is shown in Figure 16.

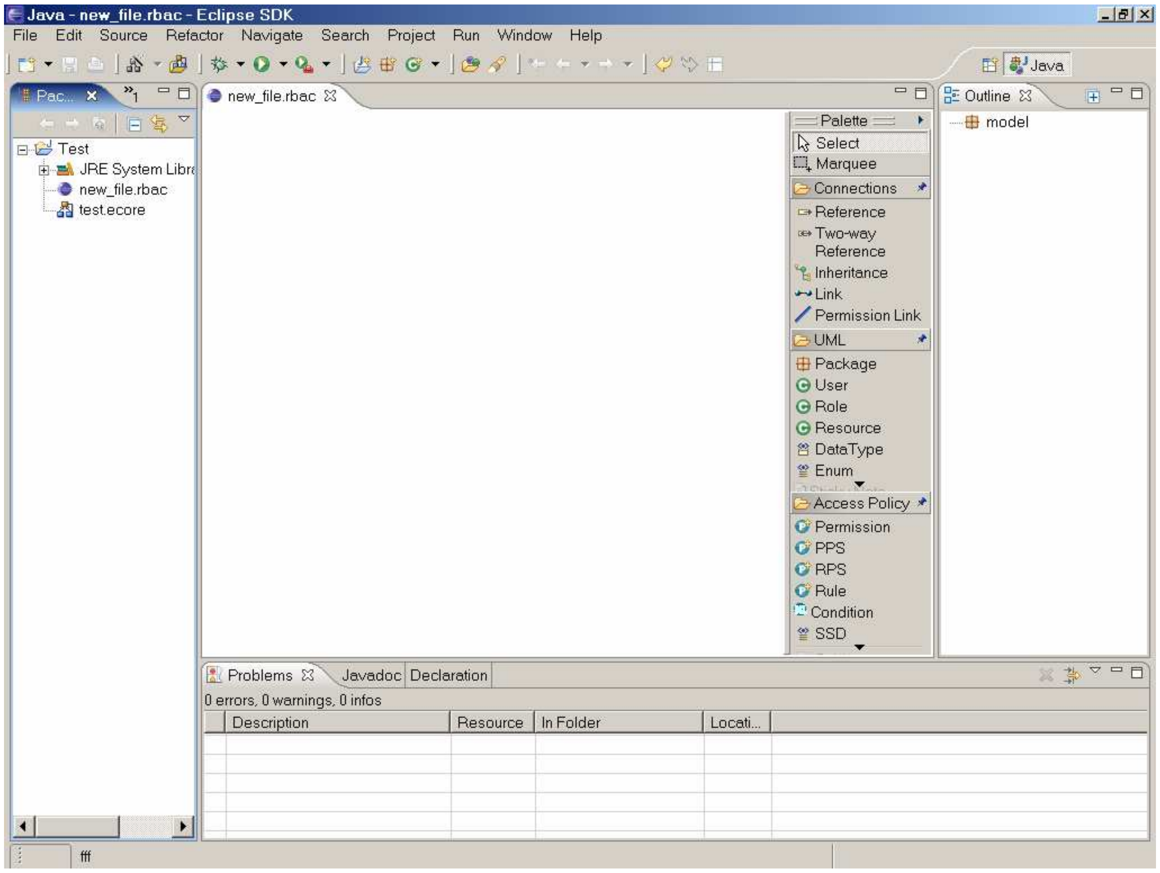


Figure 16: RBAC XACML Modeler

It has a diagram palette with icons for all supported models. The user can click an icon to draw the corresponding model element, (e.g. a User Class). Details about creating each model element are described below:

- Model User Classes, Role Classes and Resource Classes.

  User classes, Role classes and Resource classes can be created by clicking the corresponding icons in the drawing palette. The public attributes and operations can be created and added into these classes. A Role hierarchy is created by clicking the inheritance icon. A screen shot of User, Role, and Resource classes creation is shown in Figure 17.

Figure 17: Modeling User, Role, and Resource Classes

- Create a Policy Component

  A Policy Component is created by clicking the Permission icon. The combine algorithm for the Policy Component is defined through the property window. A Policy Component must connect with at least one Resource Class. A screen shot of creating a Policy class is shown in Figure 18 (nexr page).

Figure 18: Creation of Policy Class

- Create a Rule component

  A Rule component is created by clicking the Rule icon. It contains the Resource Element and Action Element, which are created by clicking the corresponding icons. Its "*positiveEffort: Boolean*" attribute is defined in the property window. Each Rule component must be connected with a Policy component. A screen shot of creating a Rule class is shown in Figure 19 (next page).

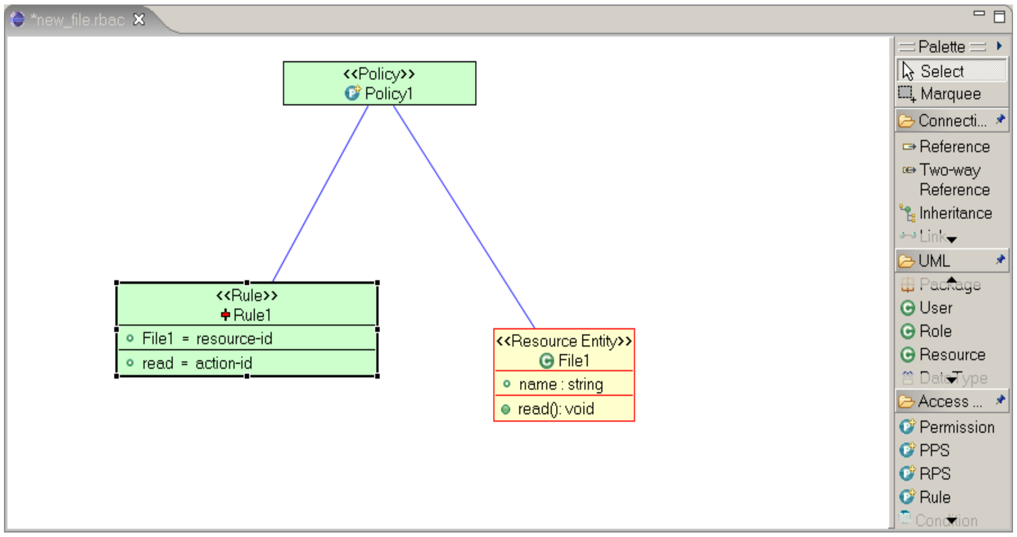


Figure 19: Creation of Rule Class

The attributes for the Resource Element and Action Element are defined in its property window. A screen shot for the Resource Element is shown in Figure 20.

Figure 20: Resource Target Element Attributes

- Create a Condition component

A Condition component is connected to a Rule component to refine the policy's applicable conditions with an OCL expression string. It is created by clicking the Condition icon. A screen shot of creating a Condition class is shown in Figure 21.

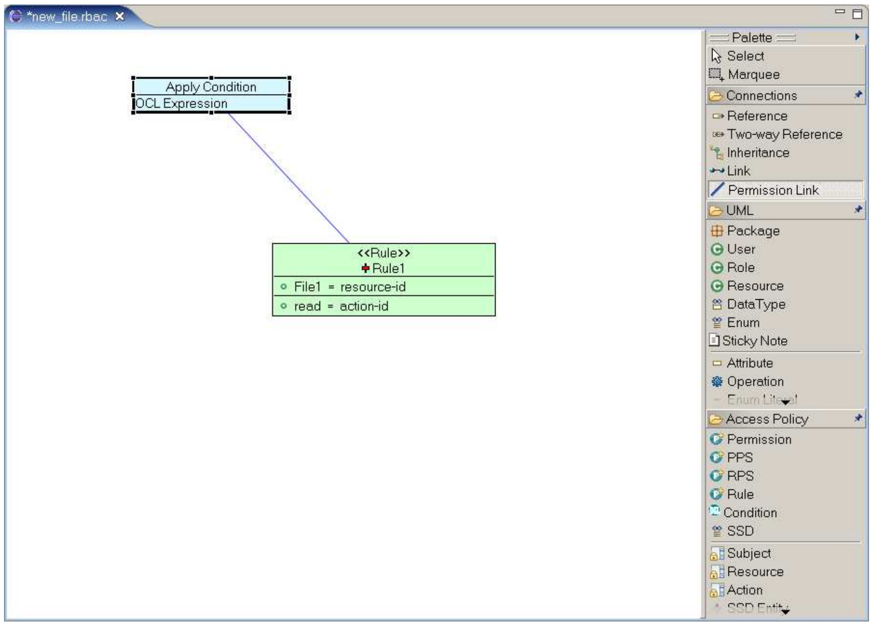


Figure 21: Condition Constraint

- Create a PPS Component and a RPS Component

The PPS Component and the RPS Component are created by clicking the PPS icon and the RPS icon. The combine algorithm for these two components is defined through the property window. A PPS component contains all applicable Policy components for a given Role. A RPS component is connected with one PPS component and at least one Role component. A screen shot of creating RPS and PPS components are shown in Figure 22 (next page).

Figure 22: Creation of RPS and PPS Classes

- User to Role Assignment

  The user to role assignment is created by clicking the “two-way references” icon to create a connection between a User class and a Role class. A screen shot of a User-to-Role assignment is shown in Figure 23.

Figure 23: User-to-Role Assignment

- Create a SSD Table

  The SSD Table is created by clicking the SSD icon. It contains a list of SSDItems, which is created as a string attribute by clicking the SSDEntity icon. A screen shot of creating a SSD table a SSD Entity is shown in Figure 24.

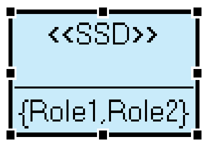


Figure 24: Creation of SSD Table

5.3 An Example

We will present a simplified distributed bug tracking application for software development as an example to demonstrate how to use the RBAC XAML Modeler to create an RBAC XACML application and generate policy specifications.

5.3.1 Problem Statement

We assume there are three roles, which are the User, the Developer and the Tester, in the bug tracking application. The application has a list of bug records. A user has permission to create a bug. A bug has an owner, an ID and a status. The status of a bug can be opened, in progress, fixed or closed. A developer has permission to change the status of a bug only if he/she is the owner of the bug, plus any permission associated with the user role. A tester has permission to close a bug only if the bug's status is fixed, plus any permission associated with the user role. However, a person cannot be the developer and the tester at the same time.

Therefore, the developer and the tester are senior to the user role, and the developer and the tester are mutually exclusive roles.

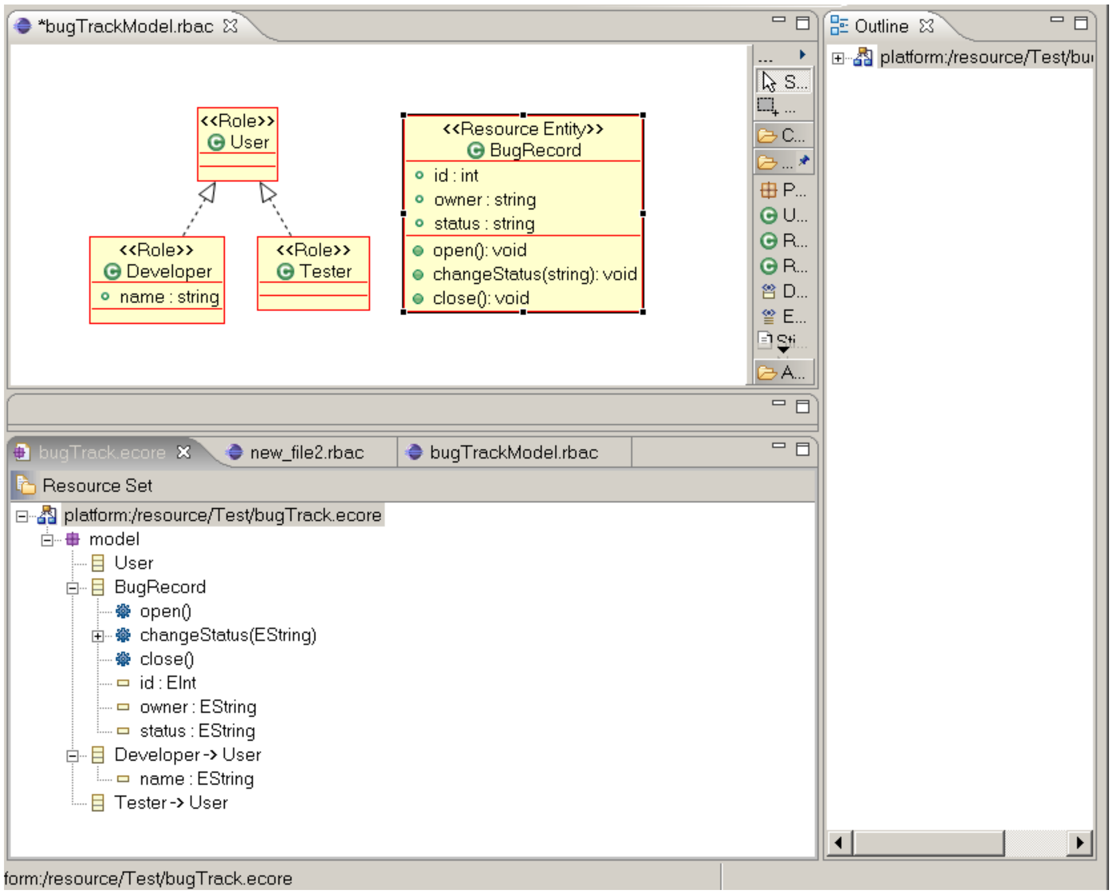
In the following sections, we are going to show to how to model this application's various roles and explain the generated policies.

5.3.2 Design and model procedures

The steps to model the bug tracking applications are as follows:

5.3.2.1 Modeling the role and resource components

According to the problem statement, there will be three role components (User, Developer and Tester) and one resource component (BugRecord). The Developer and Tester components inherit the User component. The BugRecord has an *id*, an *ownerName*, and a *status* attribute. It also has *open*(), *changeStatus*() and *close*() operations. The RBAC XAML Modeler creates an EMF model automatically for these components. A screen shot for User, Developer, Tester and BugRecord classes, and the generated EMF model is shown as Figure 25 (next page).



```
platform:/resource/Test/bugTrack.ecore
  model
    User
    BugRecord
      open()
      changeStatus(EString)
      close()
      id : EInt
      owner : EString
      status : EString
    Developer -> User
      name : EString
    Tester -> User
form:/resource/Test/bugTrack.ecore
```

Figure 25: The bugTrack Ecore model

5.3.2.2 Modeling policy specifications

According to the problem statement:

- The User role has an RPS component, a PPS component, a Policy component that contains a Rule component where the resource target is the BugRecord, and the action target is the *open*() action.
- The Developer role has an RPS component, a PPS component, a Policy component that contains a Rule component where the resource target is the BugRecord, and the action target is the changeStatus() action. An additional condition is that that the Developer must be the owner of the BugRecord.

- The Tester role has an RPS component, a PPS component, a Policy component that contains a rule component where the resource target is the BugRecord, and the action target is the *close*() action. An additional condition is that the status of the BugRecord is fixed.

The model is shown in Figure 26.

Figure 26: Policies

The RBAC XACML Modeler generates a permission policy file and a role policy file for each role. The generated files are included in the Appendix.

5.3.2.3 Modeling the role assignment policy specification and SSD

To demonstrate the user-to-role assignment, we assume there are three users, who are user1, user2 and user3. User1 is assigned to the Tester role. User2 is assigned to the User role.

User3 is assigned to the Developer role. The Developer role and the Tester role are two mutually exclusive roles, which would be defined in an SSD table. The model is shown in Figure 27.

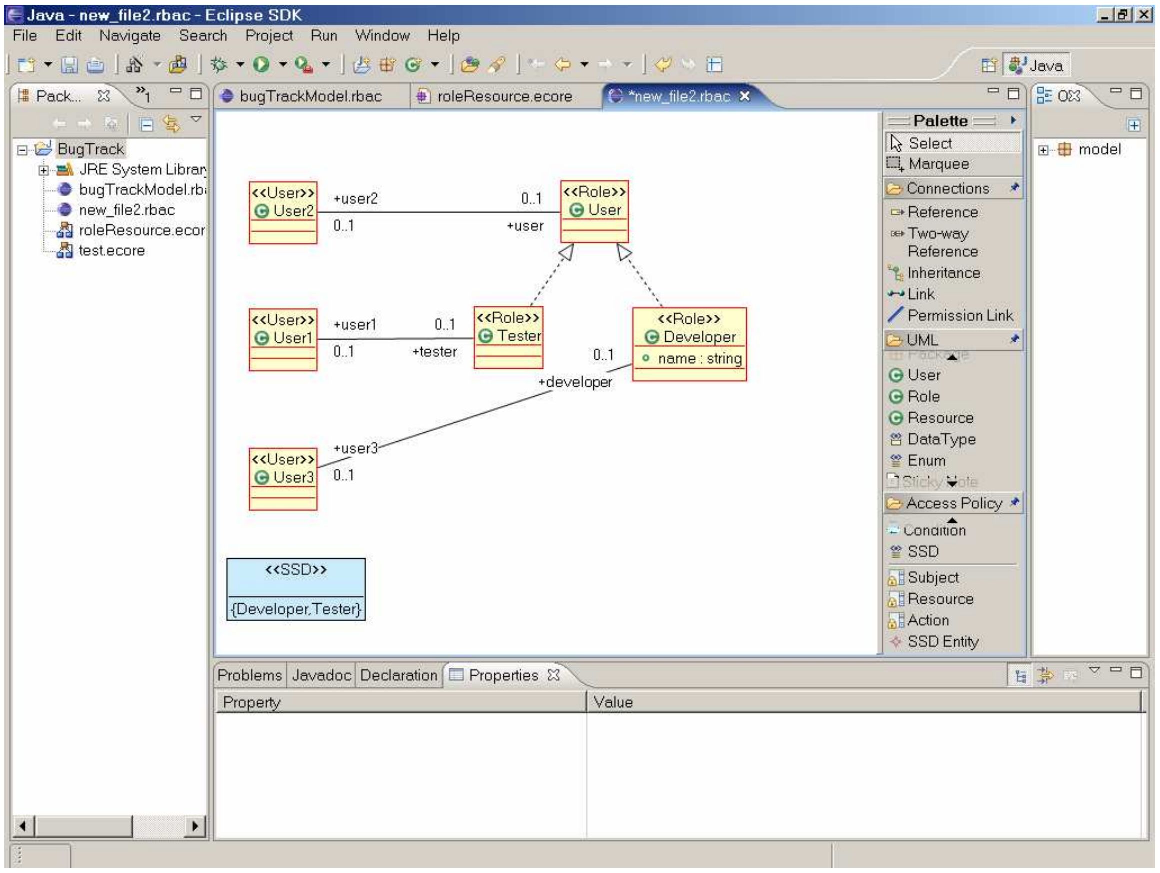


Figure 27: User-to-Role assignment and the SSD Table

The RBAC Modeler generates a role assignment policy file which indicates the user-role assignments and in which the SSD checking is defined as a condition. The condition denies the role assignment request if the applying subject already has a Developer role or a Tester role enabled, and is applying to enable a Developer role or a Tester role in the request. The file fragment representing the condition part is shown in Table 25. The generated role assignment policy file is included in the Appendix.

```
   <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
<Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">developer</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">tester</AttributeValue>
       </Apply>
       <SubjectAttributeDesignator                                    AttributeId="activedRole"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </Apply>
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
<Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
         <SubjectAttributeDesignator                                    AttributeId="applyRole"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
       </Apply>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">developer</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">tester</AttributeValue>
       </Apply>
     </Apply>
   </Condition>
```

Table 25 : Generated condition file fragment

5.3.2.4 Policy specifications validation

The RBAC Modeler relies on the SUN XACML to guarantee that the generated policy specifications always have correct schemas. RBAC Modeler also provides the policy context checking functions to help software designers avoid design errors during the policy specification designs.

- The RBAC Modeler checks the user-to-role assignment against the SSD table to avoid the conflict assignments. If a designer creates a connection between a user class and a role class that is exclusive to one of the roles that is already connected with the user class, an error message will be displayed and the connection will not be created.

- The RBAC modeler also checks the attribute values in the resource target match, the action target match, and the apply condition expressions against the role and resource classes, to ensure these attribute values are actual attributes or methods of the corresponding role or resource classes. For example, in the bug tracking application, if a designer defines an action target match for the *getStatus*() action in a rule component whose resource target is the BugRecord class, a warning message will be displayed to tell the developer that there is no *getStatus* method defined in the BugRecord class. Therefore, this rule is redundant because this action target match will never be successful.

By checking the role assignment against the SSD table and checking the target attribute values against the role and resource classes, RBAC Modeler helps the software designer to improve the correctness of policy specification designs.

Chapter 6 Contributions and Future Work

This chapter reviews the contributions of this thesis and discusses the issues that need to be addressed in the future work.

6.1 Contributions

We presented an MDA driven approach for the policy specification design of RBAC systems. We focus on visually modeling an RBAC system to fulfill the XACML profile and automatic generation of the security infrastructure in XACML-format documents.

The key contributions are listed below:

- Use the MDA approach to integrate the policy administration with the application design process. [4.2][4.3][4.4]

  We presented a platform independent metamodel for the RBAC XACML application, defined a UML profile for this metamodel, shown how to construct a RBAC model with the defined UML profile and how to map this model to a platform specific model, and finally automatically generate policy specification documents in XACML format.

- Introduce how to use OCL to express functions that are defined in the XACML Version 2.0. [4.4.2.2]

  We defined mappings between the OCL expressions and the predefined functions in XACML standard, therefore, the OCL expressions can be used along with the UML notations to model the apply condition blocks and the target element match blocks in XACML architecture.

- Extending the RBAC XACML Profile to support the SSD RBAC constraints with Role Assignment Policies.[4.2.3]

  We demonstrated how to describe the SSD constrains rules in the Role Assignment Policies; therefore, the RBAC XACML Profile standard, which is defined only for core and hierarchy RBAC models, is extended to support the SSD constraints model.

- Designing and implementing a prototype modeling tool RBAC XACML Modeler. [Chapter 5]

  RBAC XACML Modeler is an Eclipse plug-in application. It is designed and implemented to demonstrate the presented MDA approach. It allows user to design high-level, visual models for the RBAC application from the beginning of the software design phase and carrying it through the entire development process, and use generative techniques to automate the construction of systems from these designs.

6.2 Future work

Several improvements need to be addressed in future work.

These improvements are:

- Adding the XPath and XQuery support in the presented metamodel.

  XPath and XQuery[30] are used widely to select specific pieces in the XML document as retrieving data from a database. In XACML standard the AttributeSelecter uses an XPath expression to specify the selection of data from an XML resource. This component is not included in the current metamodel, which can be address as future work to cover the complete XACML components.

- Complete the mapping of the OCL expressions to the XACML predefined functions.

  The mappings between OCL expressions and XACML predefined functions define in this thesis only covered a subset of the XACML predefined functions. The complemented mappings are targeted in the future work.

- Enhance the RBAC XACML Modeler.

  Some improvements can be added to the RBAC XACML Modeler. For example, instead of transforming the User, Role, Resource elements to an EMF model, and leaving users to define their own applications base on it, in future work, we can transform them in Web applications or Web services which may bring more benefits to users.

- Apply the same approach to other access control models.

The MDA approach can be applied to other access control models. The RBAC XACML Modeler can be extended to model other access control models, and automatically generate different kinds of access control policy specifications.

References

[1] Brewer, D.F.C. and M.J. Nash. *The Chinese wall security policy*. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*. p. 206-214. May 1989

[2] Bell, D.E. and L.J. LaPadula, *Secure computer systems: Unified exposition and multics interpretation*. 1976, The Mitre Corporation.

[3] Ferraiolo, D. and R. Huhn. *Role-Based Access Control*. In *Proceedings of 15th National Computer Security Conference*

[4] *American National Standard for Information Technology-Role Based Access Control*. 2004, ANSI INCITS 359-2004.

[5] Chandramouli, R. *Application of XML Tools for Enterprise-Wide RBAC Implementation Tasks*. In *5th ACM workshop on Role-based Access Control*. p. 11 - 18. ACM Press. 2000

[6] Giuri, L. *Role-based access control in Java*. In *Proceedings of the third ACM workshop on Role-based Access Control*. p. 91-100. ACM Press. 1998

[7] Damianou, N., et al. *The Ponder policy specification language*. In *Workshop on Policies for Distributed Systems and Networks*. p. 18-39. Springer-Verlag LNCS 1995

[8] Jajodia, S., P. Samarati, and V.S. Subrahmanian. *A logical language for expressing authorizations*. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*. p. 31-42. IEEE Computer Society Press. May 1997

[9] Ray, I., et al. *Using UML To Visualize Role-Based Access Control Constraints*. In *Proceedings of the ninth ACM symposium on Access control models and technologies*. p. 115-224. ACM Press

[10] Organization for the Advancement of Structured Information Standards. *Core and hierarchical role based access control (RBAC) profile of XACML v2.0*. 2005 [cited 2005 Nov.]; Available from: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf.

[11] Organization for the Advancement of Structured Information Standards. *XACML 2.0 Core: eXtensible Access Control Markup Language (XACML) Version 2.0*. 2005 [cited 2005 Nov.]; Available from: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[12] Object Management Group. *Unified Modeling Language Version 2.0*. [cited 2005 Nov.]; Available from: http://www.uml.org/#UML2.0.

[13] Object Management Group. *UML 2.0 OCL Specification*. [cited 2005 Nov.]; Available from: http://www.omg.org/cgi-bin/doc?ptc/2005-06-12.

[14] Fuentes-Fernández, L. and A. Vallecillo-Moreno. *An Introduction to UML Profiles*. UPGRADE, 2004. **Vol.V** (No.2): p. 6-13.

[15] Object Management Group. *MDA Guide Version 1.0.1*. [cited 2005 Nov.]; Available from: http://www.omg.org/mda/specs.htm.

[16] Object Management Group. *MetaObject Facility (MOF) Specification, Version 1.4*. [cited 2005 Nov.]; Available from: http://www.omg.org/technology/documents/formal/mof.htm.

[17] Object Management Group. *UML Profile for CORBA*. [cited 2005 Nov.]; Available from: http://www.omg.org/technology/documents/ modeling_spec_catalog.htm#UML.

[18] Eclipse Foundation. *Eclipse Platform*. [cited 2005 Nov.]; Available from: http://www.eclipse.org.

[19] Moore, B., et al., *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*. 2004: IBM Corp.

[20] Eclipse Foundation. *Eclipse Modeling Framework (EMF)*. [cited 2005 Nov.]; Available from: http://www.eclipse.org/emf/.

[21] Eclipse Foundation. *Graphical Editing Framework (GEF)*. [cited 2005 Nov.]; Available from: http://www.eclipse.org/gef/.

[22] Eclipse Foundation. *The Eclipse Modeling Framework (EMF) Overview*. [cited 2005 Nov]; Available from: http://www.eclipse.org/emf/docs.php?doc=references/overview/EMF.html.

[23] Budinsky, F., et al., *Eclipse Modeling Framework*. 2003: Addison Wesley Professional.

[24] The University of Murcia *UMU-XACML-Editor* [cited 2005 November]; Available from: http://xacml.dif.um.es/.

[25] Basin, D. and J.U. Doser. *Model Driven Security: from UML Models to Access Control Infrastructures*. In *5th International School on Foundations of Security Analysis and Design*. FOSAD. Sep.2005

[26] Fink, T., M. Koch, and K. Pauls. *An MDA approach to Access Control Specifications Using MOF and UML Profiles*. In *Proceedings 1st International Workshop on Views On Designing Complex Architectures*. Sep.2004

[27] Sandhu, R.S., et al., *Role-Based Access Control Models*. Computer, 1996. **29**(2): p. 38-47.

[28] Sun Microsystems Laboratories. *Sun's XACML Implementation*. [cited 2005 Nov]; Available from: http://sunxacml.sourceforge.net/. Only a subset of XACML

functions have been covered in this table, some functions such as date or math functions are not included in this table.

[29] World Wide Web Consortium. *XQuery 1.0 and XPath 2.0 Formal Semantics*. [cited 2005 Nov.]; Available from: http://www.w3.org/TR/2005/CR-xquery-semantics-20051103/.

[30] Ferraiolo, David F., D. Richard Kuhn and Ramaswamy Chandramouli , Role-Based Access Control 2003: Artech House.

Appendix

===

Generated Permission <PolicySet> for User:

```
<PolicySet PolicySetId="pps:User:role" policyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:ordered-permit-overrides">
  <Description>This is a generated policy set</Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <Policy PolicyId="policy1" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides">
    <Description>This is a generated policy </Description>
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule RuleId="Rule1" Effect="Permit">
      <Target>
        <Subjects>
```

```
          <AnySubject/>
        </Subjects>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">BugRecord</AttributeValue>
              <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">open</AttributeValue>
              <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
</PolicySet>
============================================================
```

Generated Role <PolicySet> for User:

```
<PolicySet PolicySetId="rps:User:role" PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:ordered-permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">User</AttributeValue>
          <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <PolicySetIdReference>pps:User:role</PolicySetIdReference>
</PolicySet>
==========================================================
```

Generated Permission <PolicySet> for Developer:

```
<PolicySet PolicySetId="pps:Developer:role" PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:ordered-permit-overrides">
  <Description>This is a generated policy set</Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <Policy PolicyId="policy2" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:ordered-permit-overrides">
    <Description>This is a generated policy </Description>
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule RuleId="Rule2" Effect="Permit">
      <Target>
        <Subjects>
          <AnySubject/>
        </Subjects>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
```

```
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">BugRecord</AttributeValue>
              <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">changeStatus</AttributeValue>
              <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
      <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <SubjectAttributeDesignator AttributeId="name"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">BugRecorder.owner</AttributeValue>
      </Condition>
    </Rule>
  </Policy>
  <PolicySetIdReference>pps:User:role</PolicySetIdReference>
</PolicySet>
==========================================================
```

Generated Role <PolicySet> for Developer:

```
<PolicySet PolicySetId="rps:Developer:role" PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:ordered-permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Developer</AttributeValue>
          <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <PolicySetIdReference>pps:Developer:role</PolicySetIdReference>
</PolicySet>
==============================================================
```

Generated Permission <PolicySet> for Tester:

```
<PolicySet PolicySetId="pps:Tester:role" PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:ordered-permit-overrides">
  <Description>This is a generated policy set</Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <Policy PolicyId="policy3" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:ordered-permit-overrides">
    <Description>This is a generated policy </Description>
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule RuleId="Rule3" Effect="Permit">
      <Target>
        <Subjects>
          <AnySubject/>
        </Subjects>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
```

```
             <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">BugRecord</AttributeValue>
             <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
           </ResourceMatch>
         </Resource>
       </Resources>
       <Actions>
         <Action>
           <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">close</AttributeValue>
             <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
           </ActionMatch>
         </Action>
       </Actions>
     </Target>
     <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
         <ResourceAttributeDesignator AttributeId="status"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
       </Apply>
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">fix</AttributeValue>
     </Condition>
   </Rule>
 </Policy>
 <PolicySetIdReference>pps:User:role</PolicySetIdReference>
</PolicySet>
=================================================================
```

Generated Role <PolicySet> for Tester:

```
<PolicySet PolicySetId="rps:Tester:role" PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:ordered-permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Tester</AttributeValue>
          <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <PolicySetIdReference>pps:Tester:role</PolicySetIdReference>
</PolicySet>
```

==

Generated Role Assignment Policy:

```
<Policy PolicyId="Role:Assignment:Policy" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
combining-algorithm:deny-overrides">
  <Description>This is a generated policy for role assignment</Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <Rule RuleId="User3:role:requirements" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">User3</AttributeValue>
            <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">enableRole</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
<Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="applyRole"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Developer</AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId="User2:role:requirements" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">User2</AttributeValue>
            <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">enableRole</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
<Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="applyRole"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">User</AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId="User1:role:requirements" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">User1</AttributeValue>
            <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">enableRole</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
```

```
<Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="applyRole"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Tester</AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId="role:ssd0:requirements" Effect="Deny">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">enableRole</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
<Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">developer</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">tester</AttributeValue>
        </Apply>
        <SubjectAttributeDesignator AttributeId="activedRole"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
```

```
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
<Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <SubjectAttributeDesignator AttributeId="applyRole"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">developer</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">tester</AttributeValue>
        </Apply>
      </Apply>
    </Condition>
  </Rule>
</Policy>
```