

# Model-Driven Security Engineering for Trust Management in SECTET

Masoom Alam, Ruth Breu, Michael Hafner

Research Group, Quality Engineering University of Innsbruck, Austria

Email: {masoom.alam, ruth.breu, m.hafner}@uibk.ac.at

**Abstract**—Service Oriented Architectures with underlying technologies like web services and web services orchestration have opened the door to a wide range of novel application scenarios, especially in the context of inter-organizational cooperation. One of the remaining obstacles for a widespread use of these techniques is security. Companies and organizations open their systems and core business processes to partners only if a high level of trust can be guaranteed. The emergence of web services security standards provides a valuable and effective paradigm for addressing the security issues arising in the context of inter-organizational cooperation. The low level of abstraction of these standards is, however, still an unresolved issue which makes them inaccessible to the domain expert and remains a major obstacle when aligning security objectives with the customer needs. Their complexity makes implementation easily prone of error. The SECTET – a model-driven security engineering framework for B2B-workflows – facilitates the design and implementation of secure inter-organizational workflows. This contribution has three objectives. First we present a high-level domain specific language – called SECTET-PL. Being part of the SECTET-framework, SECTET-PL is a policy language influenced by Object Constraint Language and interpreted in the context of UML models. We then detail the Meta Object Facility based metamodels for the integration of business requirements with the security requirements. Finally, using Model Driven Architecture paradigm, we describe the transformation of high-level security models to low-level web services standard artefacts with the help of Eclipse Modelling Framework and OpenArchitectureWare.

**Index Terms**—Domain Specific Language, Model Driven Architecture, Model Driven Engineering, Service Oriented Architecture

## I. INTRODUCTION

Service Oriented Architectures (SOAs) with underlying technologies like web services and web services orchestration have opened the door to a wide range of novel application scenarios, especially in the context of inter-organizational cooperation. By providing a solution to interoperability concerns, the advent of web services standards [1] have paved the way for the integration of business processes. Due to this flexibility, on the one hand, business partners having common business goals can connect their enterprise applications regardless of their platform or technology in use. On the other hand,

this decentralized management has increased the exposure of enterprise applications and requires a thorough investigation of their security implications. Access control and privacy issues are always of major concern to distributed applications.

Applications built on web service technologies use plenty of standards like WS-Trust for trust negotiation, WS-security for the fulfilment of security requirements like confidentiality and integrity and eXtensible Access Control Markup Language (XACML) for the specification of access policies to name a few. These open standards enable the agreement and inter-operability at the technical level, upon which different proprietary applications can be built. The low level of abstraction of these standards is, however, still an unresolved issue which makes them inaccessible to the domain expert and remains a major obstacle when aligning security objectives with the customer needs. Their complexity makes implementation easily prone of error. Further, these web services security standards have a distinct *inter-operability oriented* focus – that is, these standards abstract the heterogeneity of the underlying middleware platforms only rather than the *design intent* – which is, to model high-level security concepts in application domains. Enormous growth of these standards is also a vexing problem. The intricate interdependencies among these standards (e.g. dependencies among XPath, XQuery and XACML etc) require a considerable amount of time in mastering these standards which could be a daunting task. A pressing need is therefore, to develop techniques that can abstract the complexities of the underlying security architectures (based on web services security standards).

Moreover, in order to provide a satisfying alignment between the high-level security concepts in application domains and its corresponding implementation, all stakeholders involved in the realization of the distributed system – from the domain experts to the software engineers – must have a common understanding of the security requirements, each one at the appropriate level of abstraction. This means that the security concepts should not be confined only either to high-level descriptions or to low-level implementations. A very strong binding between the high-level security models and the underlying implementation is required and only abstraction techniques are not enough.

Over the years, an important goal of the software researchers is to develop techniques to model domain con-

This paper is based on “Modeling Permissions in a (U/X)ML World” by M. Alam, R. Breu, and M. Hafner, which appeared in the Proceedings of International Symposium on Frontiers in Availability, Reliability and Security (FARES), in conjunction with The First International Conference on Availability, Reliability and Security, ARES 2006 © 2006 IEEE.

cepts in terms of their design intent rather than the underlying implementation environment. Effective complexity management techniques play a significantly important role in the development of accurate, reliable, and maintainable information systems which become increasingly larger, complex and distributed in nature. In this context, *Model-Driven Engineering* (MDE) is a promising approach 1) that treats models as an important artefact during software development; 2) that precisely envisages the problem and the solution domain at different level of abstractions and; 3) that defines methodologies for each level of abstraction and provides techniques to lower the level of abstraction by defining relationships between the participating models. The approach of MDE has two pillar aspects. The aspect *Domain Specific Language* (DSL) helps to model concepts in specific application domains such as online e-government, health-care services etc. Domain specific languages are formalized using metamodels which are used to describe relationships among concepts in a domain. The aspect *Transformation Engine* analyzes various aspects of the models in the problem domain and then synthesizes implementation artefacts from the models of the problem domain.

The software design framework from the OMG [2] – called *Model Driven Architecture* [3] – is considered as an implementation of the MDE [4]. Using the MDA framework, the software functionality is modelled with a standard modelling language (UML) as a *Platform Independent Model* (PIM) and then transformed to one or more *Platform Specific Models* (PSMs) or other PIMs. The MDA paradigm considers models an essential part of the definition of the software rather than a visual aid for understanding and communication. This makes it the most appropriate solution to abstract the complexities of the underlying platform while remaining expressive to model complex business scenarios.

We specialize the concept of MDE to *Model-Driven Security Engineering* (MDSE) by providing a framework in which security concepts in an application domain are modelled using UML and DSLs at the PIM abstraction level and are merged with business requirement models. These security enhanced PIMs are transformed to different open standard specifications (PSM) which in turn configure our component based reference architecture [5].

In summary, our contributions include (1) focusing on the security concepts in a domain at the PIM abstraction level and making a clear separation between the security models and the underlying security architectures; (2) providing an abstract representation of the underlying security artefacts responsible for the configuration of security components rather than relying on direct code generation; (3) resolving model dependencies while integrating security models with the business requirement models not only at the model and application level but at the metamodel level as well and; (4) providing explicit high-level transformations for the platform independent DSL and its supporting models to platform specific security models.

The rest of the paper is structured into two parts: In the first part, section II presents the conceptual foundations of our framework and the DSL in the context of a case study from the health-care domain. In the second part, section III elaborates two parts of our domain architecture: the metamodels (section III-A) which serve as a common syntactical and semantic base and the transformation's workflow (section III-B) through which high-level security models are transformed to low-level security artefacts. Section IV summarizes the related work and finally, in section V a conclusion is drawn.

## II. CONCEPTUAL FOUNDATIONS

A significant characteristic of the MDE is the ability to generate deliverable products from models. Using the DSLs, the approach of MDE has made it possible to build larger, more accurate, reliable and maintainable software systems. When using MDA standards, one of the possible ways for creating a DSL is to base the definition of the DSL directly on the *Meta Object Facility* (MOF) [6] – also termed as heavyweight extensions. In this way, MOF provides a common syntactical and semantic base for the DSL and other conceptual elements in the domain. A DSL can range from something that is very specific such as elements of a radar systems to something as broad as component-based middle ware applications built using platforms such as EJB, CCM [7], [8] or web services.

The SECTET project cluster – a (model driven) security engineering framework – facilitates the design and implementation of secure inter-organizational workflows. Based on the SOA paradigm, the objective of the SECTET-framework is to design and implement inter-organizational workflows in a peer-to-peer environment – i.e. without central control. Case studies from the domain of health-care, e-government and education gave us the opportunity to the apply the SECTET-framework in real life scenarios [9]–[15]. The framework weaves the ideas about MDA, MDE and web services standards together in to an inter-organizational workflow conceptual framework that is more than the sum of its parts.

The framework caters the needs of a broader domain termed as "*Security Critical Inter-organizational workflows Scenarios*". All component based application domains are under the scope of SECTET-framework broader domain. We believe that security concepts like authentication, authorization etc are fundamentally same for different application domains [16]. For example, the fundamental concept for authorization – called Role Based Access Control (RBAC) [17] is a general security concept and has been specialized in different application contexts. Similarly, SECTET-framework represents a high-level repository of security concepts which are realized in specific, component based application domains. In this sense, the abstract languages defined within the SECTET-framework broader domain are termed as DSLs. Security requirements such as workflow security requirements, access control etc are modelled using DSLs at the design

level and seamlessly integrated as security patterns in the business requirements models.

The **Trust Management** module within the SECTET-framework – the focus of this paper – deals with the realization of dynamic security requirements<sup>1</sup>. The module is composed of two major components. The objective of the **Modelling Component** within the Trust Management module is to make a clear separation of concerns at each appropriate level of abstraction and integrate them using appropriate modelling techniques. At the PIM abstraction level, the component models domain concepts with a UML profile called SECTET-UML. The profile is used to model business requirements such as data type and static security requirements such as roles and their respective hierarchies [17]. In order to model dynamic security requirements, SECTET-UML is combined with a DSL – called SECTET-PL.

SECTET-PL [11], [18], [19] – a predicative language in Object Constraint Language (OCL) style [20] – is tightly integrated with the UML – a de-facto standard for modelling. UML model property check at the PIM abstract level is the guiding principle for the design of SECTET-PL. Due to its restricted semantic scope, the generation of dynamic security policies from the abstract SECTET-PL specifications is very intuitive. Using the SECTET-PL predicates, positive and negative permissions can be specified with respect to any UML class diagram. Due to this flexibility, the dynamic security requirements specified via SECTET-PL can be transformed to any middle-ware, object-oriented security platform. SECTET-PL has been successfully applied (with some extensions) in multiple dynamic security requirement specification domains such as attribute-based delegation of rights [21] and privacy-enhanced access control [15]. At the PSM abstraction level, the Modelling Component defines domain specific web service security standard metamodels which are described in section III.

The **Transformation Component** within the Trust Management module incorporates two transformation patterns. The **Model-To-Model** (M2M) pattern deals with the transformation of high-level access requirement models to low-level XACML policy metamodel using the operational transformations of the **Query View Transformations** (QVT) [22] – an MDA standard. The **Model-To-Code** (M2C) pattern uses the XACML policy metamodel instances to generate XACML policy files. After the elucidation of the main definitions of the problem domain, the following sections endure highlighting the SECTET-UML and the SECTET-PL with an example from medical domain.

### A. Definitions of the problem Domain

Each **Partner** in an application workflow is conceived either as a **Service Provider** or as an **Attribute Authority**. In particular, the Partner which possesses the resource

<sup>1</sup>Security requirements like access control, privacy and delegation of rights that are dependent on the calling subject profiles, accessed object attributes or current system state.

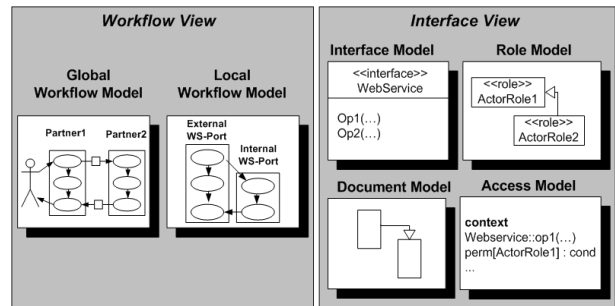


Figure 1. Model Views

and make it available as a (web) **Service** is termed as a **Service Provider** and the Partner with which the attributes of a **Service Requestor** are clarified is called an Attribute Authority. A **Global Workflow** specifies the message exchange protocol between the Service Provider and the Attribute Authority in a distributed environment. Two types of **Model Views** are used to depict the Global Workflow (cf. Fig 1). Generally, the sub-models within the **Workflow View** depict the message exchange protocol between the cooperating partners with a focus on security requirements like confidentiality, integrity and non-repudiation. In particular, the **Global Workflow Model** (GfWM) specifies message exchange between the cooperating partners and the **Local Workflow Model** (LfWM) describes the application and workflow logic.

Through the **Interface View**, each Partner contributes services to the outside with a given data type and authorization requirements. By specifying a minimum set of requirements, the Interface View describes every Partner's services independent of their usage scenario.

### B. SECTET-UML – A UML Profile for Trust Management

We conceive an application workflow as a network of partners cooperating in a controlled way by calling services and exchanging documents. The Model Views are used to depict a specific application domain scenario by specifying the messages each Partner exchanges and services which every one of them agreed to contribute to the workflow with additional constraints (security requirements) [10], [12], [13]. As Trust management is the main focus of this paper, we concentrate on the Interface View only.

Table 1 shows the stereotypes used to map the model elements of the Interface View to their representation in UML (cf. Fig 2,3a,3b).

Table 1. Model Elements of the Interface View Mapped to UML Stereotypes

Interface View Model Type	UML Model Element	UML Stereotype
Interface View	Package	<<interfaceView>>
InterfaceModel	Package	<<interfaceModel>>
Interface	Interface	<<interface>>
RoleModel	Package	<<roleModel>>
TechnicalRole	Class	<<partnerRole>>
HumanRole	Class	<<domainRole>>
DocumentModel	Package	<<documentModel>>
UniqueAttribute	Attribute	<<key>>
DistributedEntity	Class	<<d>>
ExternalFunctions	Interface	<<external>>
AccessModel	Package	<<accessModel>>
AccessRules	Constraint	<<accessConstraint>>

In the Interface View, the sub-model **Document Model** describes a data type view for the documents travelling

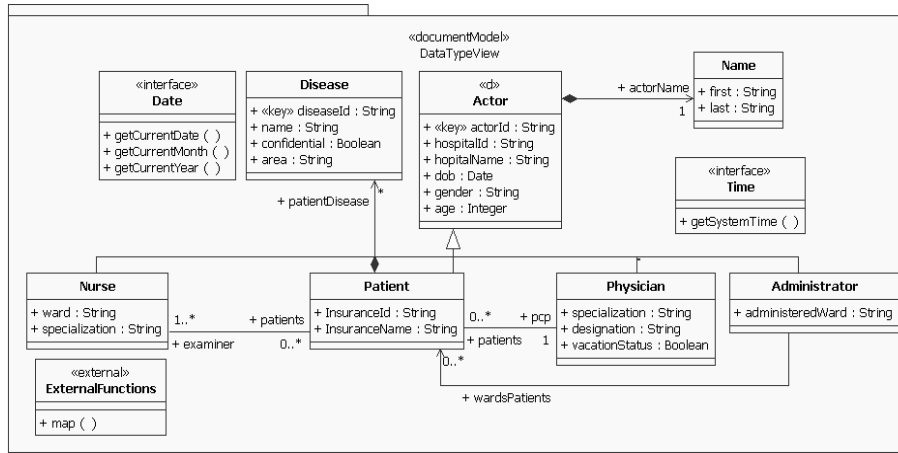


Figure 2. Sample Document Model

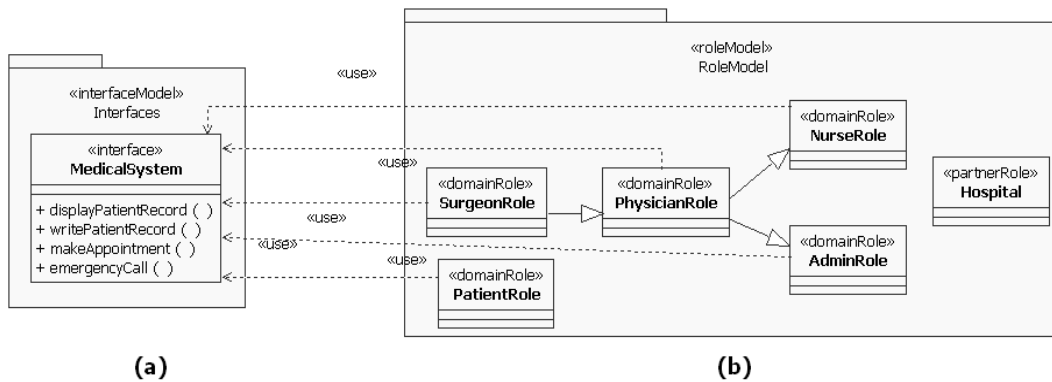


Figure 3. Sample (a) Interface Model (b) Role Model

between the partners and for the profiles of the users in the form of a UML class diagram. We talk of documents in order to stress that we do not interpret this class diagram in the usual object oriented setting but in the context of XML schema. Figure 2 shows an example Document Model of the `MedicalSystem` web service focusing only on the profiles of the users. Some classes in the Document Model have a stereotype `<<d>>`. This stereotype indicates that attribute values in the corresponding entity (e.g. patient data) can be possessed by some other Partner (or hospital) in the workflow. Attributes and entities that are not available locally will be requested by an attribute requesting service from the corresponding Partner.

Some entities in the Document Model contain an attribute which has a stereotype `<<key>>` e.g. `Physician`, `Disease` etc. This stereotype indicates that the corresponding attribute is unique and serves as a primary key for the entity. This key attribute can also be used during navigation to locate a particular object of the entity. The class `Actor` generalizes the callers of the web service and all actor classes e.g. `Physician`, `Patient` etc are specializations of `Actor`. The Document Model also contains some *context function* classes e.g. `Time`. These classes contain query functions which give date, time or some domain specific values. The inter-

face `ExternalFunctions` defines *external functions* which verify the relationship between the caller of the web service and particular elements of the Document Model. External functions will be presented in section II-C.

In the Interface View, the sub-model *Interface Model* defines an abstract set of UML operations the component offers to its clients. Figure 3a shows the Interface Model of the `MedicalSystem` web service. The data types of the parameters in the operations of the web service (interface) are either basic data types or classes in the Document Model. Hence, the Document Model provides a data type view not only for the attributes of the callers of the web services but also for the documents travelling between the partners in an inter-organizational workflow. The Interface Model together with the Document Model is used to create a Web Service Description Language (WSDL) [23] file for describing interfaces to web-facing components [24].

The sub-model *Role Model* (cf. Fig 3b) describes roles having access to the (web) services and is expressed as a UML class diagram. The Role Model is used in the *Access Model* to aggregate the rights of a particular role. In our scenario, `Hospital` represents a Partner role (stereotyped `<<partnerRole>>`) and all sub roles specific to the domain of `Hospital` e.g. `NurseRole`, `AdminRole` etc are represented domain roles (stereotyped `<<domainRole>>`). Hence each role

TABLE I.  
SECTET-PL

Navigation Expressions	
	Let $X$ be a set of typed variables. The set of typed navigation expressions over $X$ is defined as follows.
N1	Each $x \in X$ of type $T$ is an expression of type $T$ .
N2	If $e$ is a navigation expression of class type $A$ and $a$ is an attribute of type $T$ in class $A$ then $e.a$ is a navigation expression of type $T$ .
N3	If $e$ is a navigation expression of class type $A$ and there is an association between class $A$ and class $B$ as depicted in Figure 4 then $e.role$ is a navigation expression of type $B$ if the multiplicity $a..b = 1..1$ , $Set(B)$ otherwise.
N4	If $e$ is a navigation expression of type $Set(A)$ , $a$ is an attribute of type $T$ in class $A$ then $e.a$ is a navigation expression of type $Set(T)$ .
N5	If $e$ is a navigation expression of class type $Set(A)$ and there is an association between class $A$ and class $B$ as depicted in Figure 4 then $e.role$ is a navigation expression of type $Set(B)$ .
N6	If $e$ is a navigation expression of type $A$ and $A \leq B$ in the class inheritance relation then $e$ is a navigation expression of type $B$ .
N7	If $f(y_1 : T_1, y_2 : T_2, \dots, y_n : T_n) : T$ is a query operation of some interface $I$ in the Document Model and $e_1, e_2, \dots, e_n$ are expressions of type $T_1, T_2, \dots, T_n$ , resp., then $I.f(e_1, e_2, \dots, e_n)$ is a navigation expression of type $T$ .
N8	If $e$ is a navigation expression of type $Set(A)$ and $a$ is an attribute of class $A$ then $e.selectOne(boolean-expression-with-a)$ and $e.select(boolean-expression-with-a)$ are navigation expressions of type $A$ ( $Set(A)$ , respectively). $e.selectOne(p)$ returns one entity of type $A$ for which the Boolean expression $p$ is true (undefined if no such entity exists), $e.select(p)$ returns all entities of type $A$ for which the Boolean expression is true.
N9	For each class $A$ in the Document Model $subject.map(A)$ is an expression of type $A$ .
Permission Predicates	
P1	If $e1$ and $e2$ are navigation expressions of the same type then $e1 = e2$ is a permission predicate. (Additionally, the usual comparison operators $<, \leq, \dots$ on expressions of type integer and real are provided).
P2	If $e$ is a navigation expression of type $Set(A)$ and $a$ is an attribute in class $A$ then $e.forAll(Booleam-expression-with-a)$ is a permission predicate.
P3	If $e1$ and $e2$ are permission predicates then $e1$ or $e2$ , $e1$ and $e2$ , $not(e1)$ are permission predicates.
P4	If $p$ is a permission predicate with variable $z$ of type $T$ and $e$ is a navigation expression of type $T$ then $let z:T = e$ in $p$ is a permission predicate.

either stands for a human being or some other system that

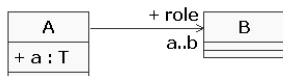


Figure 4. Association with Role

authenticates to the given node (where the way how actors authenticate can be freely chosen). Notice the difference between `PhysicianRole` in the Role Model standing for the calling physician and the class `Physician` in the Document Model standing for the internal representation of the physician’s data in the Partner node.

The sub-model Access Model extends each operation definition (in the Interface Model) with a permission section. This section specifies rules and conditions under which a given role in the Role Model is permitted (or prohibited) to call a specific web service of the Interface Model. These rules are described in the predicative language SECTET-PL which is described in the next section.

C. Predicative Specification of Access Rights in SECTET-PL

SECTET-PL is a predicate language in OCL style allowing the specification of fine-grained data dependent access permissions based on roles. Originally developed with the goal of integrating aspects of authorization in use case based development [25] we use this language in the SECTET context to specify permissions for calling web services. The approach is model driven in the sense that we generate platform independent XACML policies from the predicative specifications (cf. Section III).

In the Access Model each operation  $op$  of the Interface Model is associated with a set of rules according to the following structure:

```

context WebService : op( $x_1 : T_1, x_2 : T_2, \dots, x_n : T_n : T$ )
...
perm[ $role_i$ ] : condExp $_i$  ... ;
proh[ $role_j$ ] : condExp $_j$ 
...;
    
```

A positive rule  $perm[role_i] : pcondExp_i$  describes the condition  $pcondExp_i$  under which some role  $role_i$  is permitted to call web service  $op$ . A negative rule  $proh[role_j] : ncondExp_j$  describes the condition  $ncondExp_j$  under which some role  $role_j$  is prohibited to call web service  $op$ . The conditions are permission predicates over the formal parameters of the web service  $(x_1 : T_1, x_2 : T_2, \dots, x_n : T_n)$ . Permission predicates allow navigation through XML documents, comparison of expressions and the connection of predicates by logical operators. The language of permission predicates is defined in Table I. In the current core version of SECTET-PL, we use a simple type system which supports basic types, class types and a set type (with usual operators like arithmetic operators, union, notEmpty, etc. which are not mentioned in Table I).

The following examples refer to the Document Model, Interface and Role Model in Figures 2,3a,3b respectively.

**Rule 1:** A person, identified by his or her patient number, is allowed to make an appointment only for weekdays.

```

context MedicalSystem:makeAppointment
(patID:String,phyID:String,Day:String)
proh[PatientRole]: Day = "Saturday" or Day = "Sunday";

```

The next example constraint depends on the internal representation of the calling actor. The underlying XML documents are provided by the parameters of the web services on the one side, and by the special construct `subject.map(T)` on the other side which is supported through a library of *external functions*. The Document Model hosts the library of external functions in the interface `ExternalFunctions` stereotyped with `<<external>>`. This stereotype indicates that the corresponding interface is not transformed to XML schema but refers to the security infrastructure in order to verify a certain relationship between the caller of the web service and a particular element of the Document Model. The *identification variables* (e.g. `subject`) associated with these external functions distinguish different types of callers, e.g. `subject.map(T)` allows the connection of the calling actor with his/her internal representation to the business logic enabling permissions like "the actor has access to his/her own data". In Rule 2, an association between the calling `Physician` and the `Patient` is required whose identity is passed as a parameter. In order to provide this connection we make use of the `subject.map` construct mapping the calling actor (in the `PhysicianRole`) to its internal representation of class `Physician`.

**Rule 2:** A physician can modify any medical record for which he or she is the designated primary care physician.

```

context MedicalSystem:writePatientRecord(patId:String)
perm[PhysicianRole]:
let ph=subject.map(Physician) in
ph.patients.select(actorId=patId).notEmpty();

```

Used in some permission or prohibition expressions, the special construct `subject.map(T)` authenticates the caller of the web service (where the way how authentication is done can be freely chosen), checks if the caller is in the specified role and maps it to an internal representation in the Document Model. In case the caller belongs to the some other domain, the `subject.map(T)` requests the attribute values that are not present locally from the corresponding domain through an attribute requesting service [15].

The next examples show the combination of positive and negative permissions.

**Rule 3:** A nurse can read the medical records of every patient for her specialization. But she shall not be permitted to read the medical records of the patient if patient disease is marked confidential.

```

context MedicalSystem:displayPatientRecord(patId:String,
patDiseaseId:String)

```

```

perm[NurseRole]:
let nr = subject.map(Nurse) in
nr.patients.selectone(actorId=patId).
patientDisease.select(diseaseId = patDiseaseId and
area = nr.specialization) .notEmpty();
proh[NurseRole]:
let nr = subject.map(Nurse) in
nr.patients.selectone(actorId = patId).
patientDisease.select(diseaseId = patDiseaseId
and confidential = true).notEmpty();

```

In general the evaluation strategy of a set of permissions and prohibitions referring to some role `role` and operation `op` is such that all permission conditions  $pcond_1, \dots, pcond_n$  and all prohibition conditions  $ncond_1, \dots, ncond_m$  are connected by a logical "or" leading to the following access condition:

$$pcond_1 \text{ or } \dots \text{ or } pcond_n \text{ and } \text{not}(ncond_1 \text{ or } \dots \text{ or } ncond_m)$$

**Rule 4:** An administrator is allowed to check the medical records in working hours only.

```

context MedicalSystem:displayPatientRecord(patID:String,
patDiseaseId:String)
perm[AdminRole]:
Time.getSystemTime()>=8 and Time.getSystemTime()<=17;

```

Taking into account inheritance relations between roles, the positive permissions in the formula above are extended by permissions referring to superior roles (again by connection through a logical "or"). This means that positive permissions are inherited while negative permissions are not inherited. The concept of inheritance of permissions in the presence of role hierarchies is extensively elaborated in our model *Constraint based Role Based Access Control* (CRBAC) [26].

Following a fail-safe defaults principle the result of the evaluation is an exception in cases where no rule can be applied (there is neither direct permission nor an inherited permission).

### III. DOMAIN ARCHITECTURE

This section explains three parts of our domain architecture. In the first part, we discuss the integration of `SECTET-UML` with `SECTET-PL` at the metalevel. The second part elaborates the transformation of platform independent models to platform specific models using QVT transformations. Finally we exemplarily show how information captured via platform specific models can be translated to executable web services artefacts.

#### A. Metamodels

A metamodel defines a schema for the specification of additional semantics of existing information [27]. The Meta Object Facility (MOF) is an OMG standard which



on the basis of their content and form the input to the decision request according to some rule-combining algorithm specified as an attribute of the <Policy> element. According to the XACML data-flow model a *Policy Enforcement Point* (PEP) acts as a security gateway at the domain boundaries (cf. Fig 7). It intercepts a request coming from the outside and queries a *Policy Decision Point* (PDP) for access rights by sending an XACML request. The PDP looks for policies that apply to the request by matching the <Target>, which defines the set of <Subject>, <Resource> and <Action> elements that must be met for a <PolicySet>, a <Policy> or a <Rule>. Authorization decisions are often based on some additional characteristics of the subject (like role) or the resource or even the system environment (e.g., date, time etc.) via attributes, which are named values of known types. A predicate is a statement about attributes, which can be evaluated to either true or false. Optionally, a <Rule> can encapsulate a <Condition>, which is a Boolean function over a subject, resource or action and environment attributes or functions of attributes. Once a policy is found and verified as applicable to a request, its rules are evaluated. If the condition evaluates to true, then the rule's effect is returned (*permit*, *deny* or *notApplicable*).

### B. Transformations

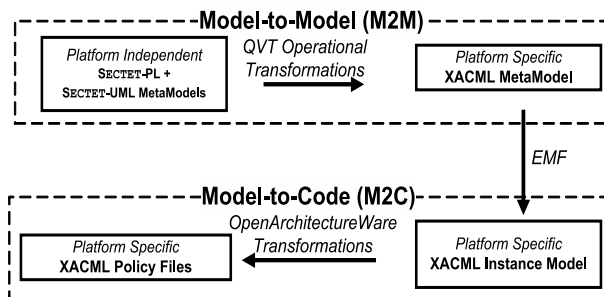


Figure 8. Transformations Workflow

*Transformations* play a key role in the MDA paradigm and are used to generate target platform specific models from the source platform independent models. The OMG has recently adopted a specification for the transformation between dissimilar metamodells called *Query View Transformation* (QVT) [22]. The QVT specification defines two flavours of transformations: *declarative* and *imperative*. In declarative transformations, the relationship between the source model and the target model is defined using the *Relations* and the *Core* language. These languages support complex object matching within source and target models and describe relationships between variables in terms of relations. In imperative transformations, an explicit sequence of steps is defined to generate a target model from the source model [22]. Figure 8 describes the whole transformation process as a workflow. The Role and the Access Model from the Domain Model are first transformed to an XACML metamodel using the

QVT operational transformation language (M2M). After M2M transformations, the XACML metamodel is populated with domain specific parts and XACML metamodel instances are generated using the Eclipse Modelling Framework (EMF) [29]. The instances of the XACML metamodel are then transformed to XACML policy files using OpenArchitectureWare's XPAND language (M2C) [30]. In the following sections, we sketch a high-level view of the transformation process particularly focusing on M2M and M2C transformations. Our goal is to provide the whole transformation process as efficiently and visibly as possible.

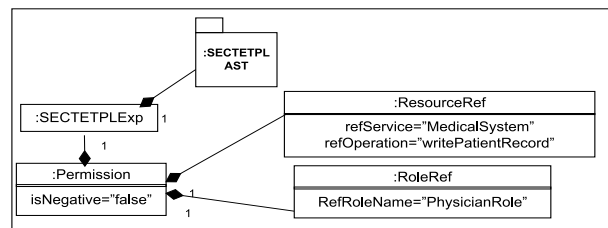


Figure 9. A Simplified Instance of Domain Model

**Model-to-Model Transformations (M2M).** Figure 9 shows a simplified instance of the Domain Model (cf. Fig 5). According to this model, the instance of the positive Permission (*isNegative=false*) is associated with a SECTETPLExp and is assigned to the instance of the RoleRef(PhysicianRole). The SECTETPLExp contains the abstract syntax tree for the constraint defined in Rule 2 (cf. section II). In this way, the Domain Model provides a common syntactical and semantic base for the SECTET-PL expressions.

```

1 transformation DomainModelToXACML (in dm: DomainModel, out policies: PolicySet)
2 main () {
3   dm.objectsOfType (RoleRef) -> map RoleRefToRolePolicySet();
4 }
5 mapping RoleRef::RoleRefToRolePolicySet() : rps:RolePolicySet
6 {
7   policySetID="RPS" + self.refRoleName;
8   policyCombAlg = "Deny-Overrides";
9   If (self.assignedPermissions.isNegative) {
10    roleReference := object PolicySetIDReference {
11      refName = "NPPS:" + self.refRoleName
12    }
13    self.assignedPermissions -> map PermissionToPPS(self);
14  } // end of if
15  self.assignedPermissions -> map PermissionToCondition();
16  } // end of mapping RoleRefToRolePolicySet
17 mapping Permission::PermissionToCondition() : cond:Condition {
18  if (not cond.setXACMLCondition (self.sectetpl) ); // Black Box Operation
19    raise "Transformation Failed";
20  } // end of mapping PermissionToCondition
21 }

```

Figure 10. Example Transformation Pattern

Figure 10 shows an example transformation pattern that uses the operational mappings of the QVT. The transformation *DomainModelToXACML* (line 1) defines two typed candidate models (as parameters): "dm" of type *DomainModel* and "policies" of type *PolicySet*.

The starting point for any operational transformation is the function `main()` – called entry operation. In this case, the `main()` function retrieves all the objects of type `RoleRef` and applies a mapping operation called `RoleRefToRolePolicySet()` to all the objects of type `RoleRef` (line 3).

The corresponding `RoleRefToRolePolicySet` mapping operation (defined at line 5) implements a mapping between one or more source model elements (`RoleRef`) to one or more target model elements (`RolePolicySet`). In general, the mapping operations populate the attributes of the target model elements with the source model element attribute values. For example, in this case `policySetID` is set to "RPS" + `self.refRoleName` where `self` refers to the instance of the source meta class and + represents concatenation.

The operational language defines an alternative method for mappings called object expression. The object expression is used to facilitate the creation and/or updation of model elements within mapping operations. For example, the object expression (defined at line 10) creates an instance of `policySetIDReference` and populates its attributes `refName` with the name of the contextual parameter `self.refRoleName`. In this way, a high-level conciseness is achieved. The mapping operation `RoleRefToRolePolicySet` also contains a call to another mapping operation (at line 12) `PermissionToPPS(self)`.

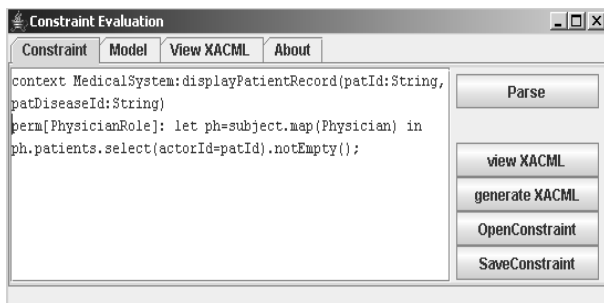


Figure 11. SECTET-PL Prototype

In order to transform the SECTET-PL expressions associated with the permission object (cf. Fig 9), we use the black box operations. According to QVT specification, black box operations can be considered as Java Native Interface (JNI) methods and are used to allow the domain specific libraries/algorithms to calculate complex model property values from the source model. The calculated values are then used to populate the target model elements. The black box operations can be coded in any programming language with MOF bindings. In this way, the specification of complex domain specific algorithms (such as syntax and semantic analysis of SECTET-PL expressions in our case) in their optimal languages can be done rather than in QVT transformation languages which are too general and therefore not best suited for coding these algorithms. Consequently, the complexity

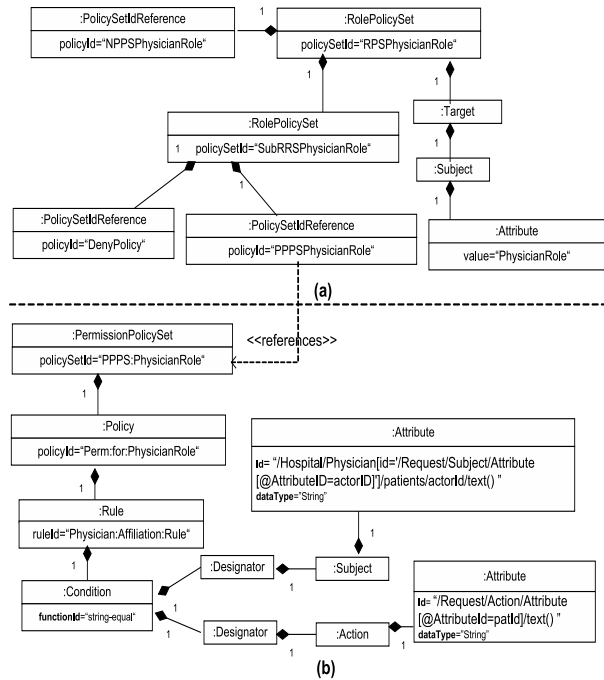


Figure 12. XACML metamodel Instance

and length of the QVT scripts is reduced significantly and makes the implementation of some parts of the transformation opaque. For instance, the mapping operation `PermissionToCondition` (line 17) contains a black box operation which returns either `true` or `false` depending on the status of the transformation. The black box operation `setXACMLCondition()` (line 18) takes an object of the AST of the `sectetpl` expressions, performs the syntax and semantic analysis and then populate the target model elements – XACML metamodel with the generated values in the form of XPath [31] expressions. The black box operation `setXACMLCondition` – implemented as a prototypical tool (cf. Fig 11) uses Antlr [32], a compiler program to generate the abstract syntax tree from the SECTET-PL expressions. The tool performs the semantic analysis of the predicates against the model information specified via XMI files.

Figure 12a shows a `RolePolicySet` instance for the XACML metamodel defined in Figure 6. The `RolePolicySet` defines a role `PhysicianRole` as a `Subject` attribute in the `Target`. It makes the `RolePolicySet` applicable to any XACML request possessing the role `PhysicianRole`. The `RolePolicySet` contains a reference to the negative `PermissionPolicySet` (generated for the `PhysicianRole`) in the `<PolicySetIDReference>` element of the XACML. The sub `RolePolicySet` contains the references to the positive `PermissionPolicySet` of the `PhysicianRole` and to a general `DenyPolicy`. In case none of the positive `PermissionPolicySet` returns `true`, the general `DenyPolicy` will be applicable.

Figure 12b shows the `PermissionPolicySet` instance of the XACML metamodel which in this case is a

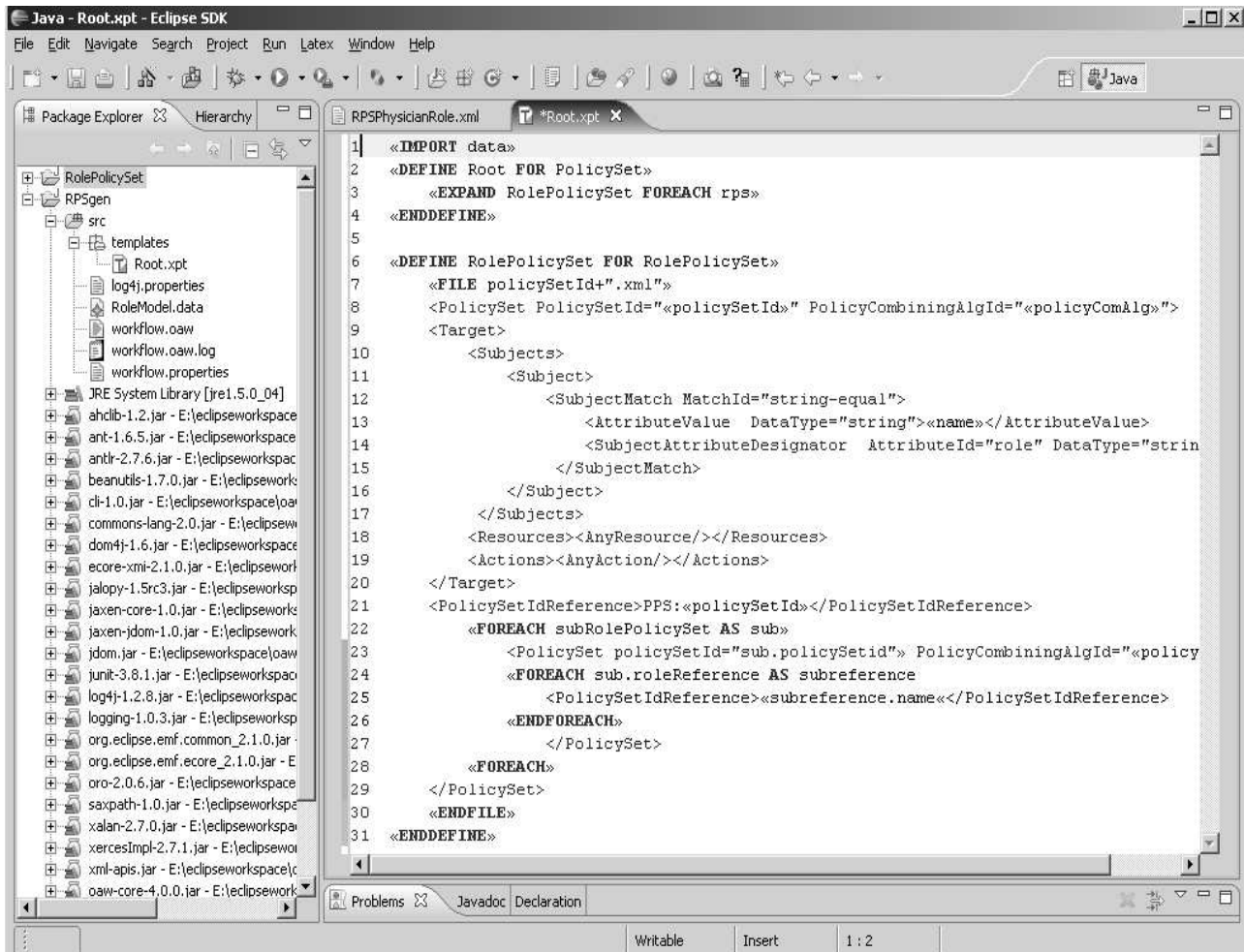


Figure 13. XPAND Language Script

positive PermissionPolicySet. This positive permission policy set contains a sub policy which defines permissions for the PhysicianRole. Within this policy, the permissions are defined as XACML rules. The target of these rules contain the web service and its operation name as Resource and Action attributes respectively. The Condition within the Rule contains the constraint and is a combination of Designator attributes (e.g. ActionDesignator) and the constraint in the form of Condition functions.

**Model-to-Code Transformations (M2C).** We use the Eclipse Modelling Framework (EMF) to generate domain specific instance models of the XACML metamodel. The XACML metamodel is drawn using Rational Rose and imported into the EMF. The imported metamodel is then populated with domain specific values and transformed to XACML policy files using the OpenArchitectureWare's XPAND template language. For more information on how it is done, please refer to [33].

Figure 13 shows an excerpt of the XPAND language script used to transform the XACML RolePolicySet instance model (cf. 12a) to XACML policy files. The script starts with an IMPORT statement (line 1) used to import the instance model packages which in this case is

data and with a definition statement (DEFINE – line 2) which defines a link to the metamodel class for which this template is defined. Within this definition statement, the EXPAND statement (line 3) defines another definition block with different variable context (rps in this case) and works as a subroutine.

Line 6 defines the corresponding definition block for the EXPAND statement defined at line 3. This definition block starts with a FILE statement (line 7) used to redirect the output from its body statements to the specified target. The attribute policySetId of the RolePolicySet instance is used as the filename (<<policySetId>>+.xml) within the FILE statement. The Subject of the Target is populated with the name attribute of the corresponding RolePolicySet instance (line 13). The FOREACH statement iterates over each element of the target instance model and generates the specified output e.g. FOREACH at line 24 iterates over all the policy references defined within the subRolePolicySet and output their names in the <PolicySetIdReference> element.

Figure 14 shows an example RolePolicySet generated for the XACML metamodel instance PhysicianRole (namespaces are omitted for

```

<PolicySet PolicySetId="RPSPhysicianRole" PolicyCombiningAlgId="Deny-Overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="string-equal">
          <AttributeValue DataType="string">RPSPhysicianRole</AttributeValue>
          <SubjectAttributeDesignator AttributeId="role" DataType="string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>
  <PolicySetIdReference>NPPS:RPSPhysicianRole</PolicySetIdReference>
  <PolicySet PolicySetId="SubRPS" PolicyCombiningAlgId="Permit-Overrides">
    <PolicySetIdReference>PPPS:RPSPhysicianRole</PolicySetIdReference>
    <PolicySetIdReference>DenyPolicy</PolicySetIdReference>
  </PolicySet>
</PolicySet>

```

Figure 14. Sample XACML Role Policy Set

brevity).

#### IV. RELATED WORK

Compared to other approaches that support a policy language for the specification of dynamic constraints, our primary goals are different from the existing approaches in that we intend to apply model-driven engineering techniques to advance aspects of access control. To the best of our knowledge, this is the first approach for model-to-model transformation of high-level access models to low-level web services models.

There are some Trust Management approaches based on **proprietary-standards** e.g. Trust-X [34] is an XML based framework for trust negotiations in peer-to-peer environments. It is intended to work in open environments like the internet where the entities have no prior knowledge of each other. Trust-X supports different strategies for trust negotiations. One of the most important features of Trust-X is to have a strategy for the ordering in which credentials should be disclosed which our modeling framework does not support at the moment. But Trust-X has no notion of modeling and its low level of abstraction makes it inaccessible for the domain expert. A major drawback of Trust-X is the lack of full support for open standards which makes policy exchange and management among different domains more difficult. Moreover, its disclosure policy is not type safe, which can cause inconsistencies in Trust Management scenarios. In [35], the authors describe a proprietary XML based framework that defines the mapping of strangers to roles based on their credentials. Their prototype called Trust Establishment (TE) does not support sensitive credentials. This means that, while assigning roles to strangers, one of the basic TE's assumptions is that credentials can be disclosed without any privacy requirements which hinder its suitability in distributed environments. KeyNote [36] is a very well-known Trust Management language. Its main feature is the support for distributed authorization through delegation policies. These policies (called assertions) contain the description of the authorizer, constraints and licensees (to whom rights are delegated). However, KeyNote language addresses only rights delegation aspect of a Trust Management scenario and the specification

of delegation policies in proprietary languages compared to open standards like XACML profile for rights delegation [37] restricts its usability. There are some other approaches like [38] which require the counterpart in a trust establishment scenario to fully rely on the syntax and semantics of their devised trust policies and hence is of restricted usability.

PolicyMaker [39] and REFEREE [40] suggest a programming language to express trust policies based on **open-standard** (digital certificates). These approaches are primarily concerned with the specification of trust policies using public key based authorization, authentication and do not consider modeling at a higher level of abstraction.

In [41], the authors have presented a **logic-based** framework to regulate service access control and information release in open environments. The framework is supported by a Portfolio and Service Protection Language (PSPL) for the specification of access policies for services and for information release. The PSPL only provides logical definition and no actual language is provided. Moreover, a prototype for the PSPL is also not available. The use of logic programming for the specification of Trust Management requirements provides a powerful mechanism to prove the correctness of the overall system but the specifications are difficult to understand and obscure transcription. Moreover, no logic-based mechanism is yet incorporated into a standard framework.

In [42], the authors have highlighted only the negotiation aspects of a Trust Management scenario in their **model-driven** framework. They use UML activity diagrams to express trust negotiations among different partners. The approach lacks the support for a high level policy language for the specification of context dependent complex trust policies.

Compared to **model-driven security** aspect of our framework, [43] presents an approach for the application of pattern-based software development to recurring problems in the domain of security. The basic idea of the approach is to capture expert-knowledge in the security domain and make it available to developers as a security pattern during software development. The approach provides an in-depth view of security patterns, its development through an ontology based knowledge base and sorting out relationships between different existing security patterns. Although the author uses patterns to systematically capture knowledge about security issues at the model level, the semantics remain close to the technical level. The author does not address transformation in any way. The author in [44] has presented a verification framework for UML models enriched with security properties through a UML profile called UMLSec. The framework stores UMLSec models using XMI files format in a Meta Data Repository (MDR) which is then queried using Java Metadata Interfaces (JMI) by different analyzers. These analyzers perform static as well dynamic analysis on the UMLSec models for different security properties like confidentiality, integrity etc. Compared to these approaches, our framework is more domain specific and focused on

the automatic generation of (standard) security artefacts specified during the early phases of software development. Further our objective is to develop high-level abstract languages through which executable security requirements of a distributed system can be specified at a higher level of abstraction. [45] proposes a UML based notation for access control using RBAC and provides code generation exclusively for object oriented platforms (J2EE or .NET). Our approach has the advantage that it can be used in any environment and also by using web services standards in our case, policy exchange and management across domain boundaries will be easier.

There are some **MDA-frameworks** that provide the means to realize metamodels and transformation through template languages. The comparison we make is based on the parameters that how much a corresponding framework is flexible to accommodate domain specific frameworks like SECTET-framework and its support for the MDA related standards. ANDROMDA [46] is an open source MDA-framework that provides metadata-handling facilities through the Apache Velocity template language. The framework uses the NETBEANS metadata repository (MDR) [47] for storing metadata and a set of cartridges for access to the MDR. A major drawback of the framework is the complexity involved in defining the domain specific extensions. Writing a new cartridge for ANDROMDA is not a very trivial job. OpenArchitectureWare is another framework that provides a more generic solution for domain specific engineering [30]. The reason is that it is open to other modelling frameworks like Eclipse Modelling Framework [29] or tools like MAGICDRAW [48]. Its template language XPAND provides an intuitive way to generate any kind of data from specified models. However, none of the above MDA-frameworks support the QVT based transformations.

## V. CONCLUSION

In this paper, we presented a model driven security engineering framework within the SECTET-framework particularly focusing on trust management issues. The primary goal of the SECTET-framework is to align the high-level security objectives of Business Services with the underlying implementation in the form of Privacy Security Trust (PST) technologies. Further, we provide the specification of dynamic constraints policies via high-level language SECTET-PL. One of the most important advantages of SECTET-PL is that it is tightly integrated with the UML models which is a de-facto standard for modelling and it combines the use of a predicative language at a high-level of abstraction with an underlying platform independent access policy standard XACML. This brings great potential for future extensions ranging from the combination and harmonization of access models to property check of access models at the abstract level. We also presented the integration of security models with the business models and generation of security artefacts using OpenArchitectureWare's XPAND language.

Currently, we are working along two lines. We are pushing an implementation of a model transformation engine based on MOF-QVT with the EMF for research purposes and extending our tool support [11], [18] to perform visible QVT transformations. Secondly, we are extending SECTET-PL for the specification of rights delegation, obligation, information filtering and separation of duty constraints.

## REFERENCES

- [1] "Web service security specifications, available at," <http://www.oasis-open.org/specs/index.php>.
- [2] "Object Management Group," <http://www.omg.org>.
- [3] "Model Driven Architecture," <http://www.omg.org/mda>.
- [4] S. Brahe and K. Osterbye, "Business Process Modeling: Defining Domain Specific Modeling Languages by Use of UML Profiles," in *ECMDA-FA 2006*, INCS 4066, pp.241-255, 2006.
- [5] M. Hafner, R. Breu, and M. Breu, "A Security Architecture For Inter-organizational Workflows-Putting WS Security Standards Together," in *ICEIS 2005, ISBN: 972-8865-19-8*.
- [6] "Meta Object Facility: OMG Adapted Specification available at, note=<http://www.omg.org/docs/ptc/04-10-15.pdf>."
- [7] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztiapanovits, and S. Neema, "Developing applications using model-driven design environments," *Computer*, vol. 39, no. 2, p. 33, 2006.
- [8] R. France and B. Rumpe, "Domain Specific Modeling," editorial of *Software System Model Journal* 2005.
- [9] M. Hafner and R. Breu, "SECTET An Extensible Framework for the Realization of Secure Inter-Organizational Workflows," in *To Appear in: Journal of Internet Research*, 2006.
- [10] R. Breu, M. Hafner, B. Weber, and A. Nowak, "Model driven security for inter-organizational workflows in e-government," in *TCGOV*, 2005, pp. 122-133.
- [11] M. Alam, R. Breu, and M. Hafner, "Modeling Permissions in a (U/X)ML World," in *IEEE ARES 2006*, ISBN: 0-7695-2567-9.
- [12] R. Breu, M. Breu, M. Hafner, and A. Nowak, "Web service engineering – advancing a new software engineering discipline." in *ICWE*, 2005, pp. 8-18.
- [13] M. Hafner, M. Breu, R. Breu, and A. Nowak, "Modelling inter-organizational workflow security in a peer-to-peer environment," in *ICWS '05: Proceedings of the IEEE International Conference on Web Services (ICWS'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 533-540.
- [14] M. Alam, M. Hafner, and R. Breu, "Model Driven Security for Web Services (MDS4WS)," in *INMIC 2004, Digi Obj Id 10.1109/INMIC.2004.1492930*.
- [15] M. Alam and M. Hafner and R. Breu, "Modeling authorization in an soa based application scenario." in *IASTED Conf. on Software Engineering*, 2006, pp. 79-84.
- [16] M. Hafner, M. Alam, and R. Breu, "Towards a mof/qvt-based domain architecture for model driven security." in *MoDELS*, 2006, pp. 275-290.
- [17] Role Based Access Control (RBAC) available at, [csrc.nist.gov/rbac/](http://csrc.nist.gov/rbac/).
- [18] SECTETPL : A Predicative Language for the Specification of Access Rights available at, <http://qe-informatik.uibk.ac.at/~muhammad/TechnicalReportSECTETPL.pdf>.
- [19] M. Alam, "Model-Driven Security Engineering for the realization of Dynamic Security Requirements in Collaborative Systems," in *PhD Symposium of IEEE/ACM Models 2006*, LNCS 4364.

- [20] "UML 2.0 OCL Specification," <http://www.omg.org/docs/ptc/03-10-14.pdf>.
- [21] M. Alam, M. Hafner, R. Breu, and S. Unterthiner, "A framework for modeling restricted delegation in service oriented architecture." in *TrustBus*, 2006, pp. 142–151.
- [22] "Query view transformation: Omg adapted specification available at," <http://www.omg.org/docs/ptc/05-11-01.pdf>.
- [23] "Web Service Description Language (WSDL), available at," <http://www.w3.org/TR/wsdl>.
- [24] "WSDL First, July 22, 2003," <http://webservices.xml.com/pub/a/ws/2003/07/22/wsdlfirst.html>.
- [25] R. Breu and G. Popp, "Actor-centric modeling of user rights." in *FASE*, 2004, pp. 165–179.
- [26] M. Alam, M. Hafner, and R. Breu, "A Constraint based Role Based Access Control in the SECTET A Model-Driven Approach," in *Proceedings of the ACM PST 2006 – International Conference on Privacy, Security and Trust, October 30th, 2006 – November 1st, 2006*.
- [27] <http://wikipedia.org/>.
- [28] "XACML 2.0 Specification Set available at," [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=xacml](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml).
- [29] <http://www.eclipse.org/emf/>.
- [30] OAW XPAND Language available at, [http://www.eclipse.org/gmt/oaw/doc/r20\\_xPandReference.pdf](http://www.eclipse.org/gmt/oaw/doc/r20_xPandReference.pdf).
- [31] XML Path Language (XPath) specifications available at, <http://www.w3.org/TR/xpath>.
- [32] <Http://wwwantlr.org>.
- [33] OAW 4 EMF Example available at, [http://www.eclipse.org/gmt/oaw/doc/30\\_emfExample.pdf](http://www.eclipse.org/gmt/oaw/doc/30_emfExample.pdf).
- [34] "Trust-x: A peer-to-peer framework for trust establishment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 827–842, 2004.
- [35] A. Herzberg, Y. Mass, J. Michaeli, Y. Ravid, and D. Naor, "Access control meets public key infrastructure, or: Assigning roles to strangers," in *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2000, p. 2.
- [36] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The keynote trust-management system version 2," , United States, 1999.
- [37] "XACML v3.0 administration policy Working Draft 05 December 2005," [http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=xacml](http://www.oasis-open.org/committees/documents.php?wg_abbrev=xacml).
- [38] T. Yu, M. Winslett, and K. E. Seamons, "Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 1, pp. 1–42, 2003.
- [39] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," Tech. Rep., 1996.
- [40] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss, "Referee: trust management for web applications," *World Wide Web J.*, vol. 2, no. 3, pp. 127–139, 1997.
- [41] P. Bonatti and P. Samarati, "Regulating service access and information release on the web," in *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2000, pp. 134–143.
- [42] H. Skogsrud, B. Benatallah, and F. Casati, "Model-driven trust negotiation for web services," *IEEE Internet Computing*, vol. 7, no. 6, pp. 45–52, 2003.
- [43] M. Schumacher, "Security engineering with patterns," INCS 2754 ISBN: 3-540-40731-6, 2003.
- [44] J. Jürjens, "Secure systems development with uml," in *ISBN: 3540007016*.
- [45] T. Lodderstedt, D. Basin and J. Doser, "A UML Based Modeling Language for Model-Driven Security ," 5th international conference UML 2002 Dresden, Germany, 2002.
- [46] AndroMDA v3.0M3 available at, <http://andromda.org/>.
- [47] "http://mdr.netbeans.org/"
- [48] "http://www.magicdraw.com/"

**M.Sc Masoom Alam** is a PhD student at the university of Innsbruck Austria. Before joining the group of Prof Dr Ruth Breu, he was a lecturer in the Institute of Management sciences Pakistan. He got PhD scholarship from the higher education commission of Pakistan for perusing his PhD abroad. The focus of his PhD is the application of model-driven techniques for the realization of complex access control patterns like restricted delegation of rights and privacy enhanced access control. His current research interests include Trusted Computing and policy model extension of Security Enhanced Linux (SELinux).

**Prof Dr. Ruth Breu** is head of the research group Quality Engineering since 2002. Before she was a researcher at the University of Passau and Technische Universität München and spent several years in industry working as consultant in the area of Software Engineering. The research group Quality Engineering with currently about 20 members focuses on novel usage scenarios of models in the areas of security engineering, IT governance and model quality assessment.

**Dr. Michael Hafner** gained industry experience in the automotive and the telecommunications sector from his former activity as a technical consultant for systems integration with one of the big five companies before joining the group of Professor Ruth Breu at the University of Innsbruck as a researcher. In his dissertation, he focused on the model driven development of secure inter organizational workflows based on Service Oriented Architectures. Michael Hafner was responsible for the design and the realization of the SECTET framework and was technical project manager in the SECTINO project.