VISUAL PROGRAMMING LANGUAGE FOR SECURITY REQUIREMENTS IN BUSINESS PROCESSES AS MODEL-DRIVEN SOFTWARE DEVELOPMENT

Mirad Zadic and Andrea Nowak
Austrian Research Centers GmbH-ARC, A-2444 Seibersdorf, Austria

Keywords: Model Driven Development, Graphical modeling environment, Security policies Architectures, Access Control Policy, Business Processes, XACM Policy Generator.

Abstract: Our approach is based on a security modeling framework and a Meta Modeling Environment for design and generating of access control and security policies for business processes. The framework introduces a methodology that focuses on both, the modeling as well as the implementation aspect of security-requirements and consists of a suite of tools that facilitates the correct realization and the cost-efficient management of decentralized, security-critical workflows. Currently, the framework is being analyzed for general suitability to domains in business processes, taking basic security requirements like confidentiality, integrity and non-repudiation. We use Model-Driven Development (MDD) approach to creating our solutions based on graphical modeling environment as EMF (Eclipse Modeling Framework), GEF (Graphical Editor Framework) and GEMS (Generic Eclipse Modeling System). This graphical modeling environment makes possible rapidly creating modeling tool from a visual language description or metamodel without any coding in third-generation languages. The framework is prototypically validated through a case study for the systematic realization of e-government related workflows. Realizations of security issues follow the steps from provide methodologies that translate the abstract security requirements into run-time artifacts for the target architecture through model transformation. On this approach for this Case study is develop a Policy Specifications modeling tool based on the metamodel describing syntax of the DSML. The important goal is the automatically generate the security artifacts (enforceable security policies in XACML format) to improve the productivity of the development process and the platform independent design. Our case study defines the Business processes, which provide secure Information between three Domains: Municipality, Environment Ministry and Registry of the Combustion plant - environmental pollution producer.

1 INTRODUCTION

In this work we present a modeling tool and generator which use evaluated modeling environment that enables developers to generate security specific Modeling Language for customizing the security requirements in security-critical inter-organizational business processes based on services-oriented architectures. As web services are often composed to carry out complex business transactions, not only the web service itself has to be secured, but also the message exchange between different web services. Web Service Security specifies a mechanism for signing and encrypting SOAP messages and this mechanism can be used to implement message integrity and confidentiality. It further supports the propagation of the authentication information in the form of security tokens (e.g., SAML, Kerberos tickets or X.509 certificates). XACML provides access control mechanisms and policies within documents, while SAML represents authentication and authorization decisions in XML format and is used to exchange this information over the internet (e.g., to support single sign-on).

Specifying security policies in XACML, especially access control policies, may be an enormous task for every security administrator. 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 the security administrator does not take part directly in the design of business process, so he/she may not understand the software structure and details well enough to define policies to fulfill the security requirements.

Many models have been developed to construct and manage the security requirements but our security infrastructure is deployed in a modular way with pre-defined, configurable security components. The goal in this work is to present the concept and design of a generator, which should support the process of specifying the access control policies. Furthermore, the generator should provide a support for security administrator and help him to tackle the well-known problems, which may arise during the security policy specification.

Model-driven software development can be seen as a new generation of visual programming languages. A visual language basically consists of two parts – domain and presentation (visual) part. The domain part of the visual language is defined by means of the domain metamodel, where the relevant language concepts and their relationships are formalized for precise definition of language semantic. Instances of classes in the metamodel are types of diagram graphical elements for a diagram definition in the presentation (or graphics) model. Other to say and very shortly express, the syntax of every model is defined by a metamodel, the model plays the role of the source code, and the generator replaces the compiler.

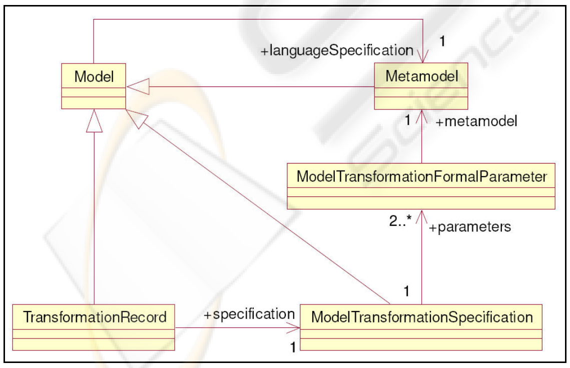

Figure 1: Core modeling process.

At the core of MDA are the concepts of models, of metamodels defining the abstract languages in which the models are captured, and of transformations that take one or more models and produce one or more other models from them. Figure 1 shows the relationships between these major concepts.

We use for metamodeling the Generic Eclipse Modeling System (GEMS), a part of the Eclipse Generative Modeling Technologies (GMT) based on MOF (*Meta Object Facility*). This framework helps developers rapidly create a graphical modeling tool from a visual language description (metamodel) without any coding in third-generation languages. The Generic Eclipse Modeling System (GEMS) substantially reduces the cost of developing a graphical modeling tool by allowing developers to focus on the key aspects of their tool: the specification of the DSML and the intellectual assets built around the use of the language. The infrastructure to create, edit, and constrain instances of the language is generated automatically by GEMS from the language specification. That make possible the separation of language development, coding, as well as “look and feel” development, such as changing how modeling elements appear based on domain analyses.

Using this approach, it is possible to generate automatically large amounts of source code and other artifacts, e.g. deployment descriptors and make reference XML files, all based on relatively concise models. This improves the productivity of the development process as well as the quality of the resulting systems. It is also a large step towards the platform independent design of systems.

2 SECURITY WORKFLOWS

The development of a security-critical inter-organizational workflow starts with the analysis and the design of the workflow, followed by a risk and threats analysis, and the security requirements specification. Security requirements are modeled in a platform-independent way at predefined levels of abstraction.

The inter-organizational business workflow is designed by representatives of the partners involved in the workflow. The security requirements are defined together with design of business workflow in each business step and there is no central control of the security requirements. The involving of security requirements in business workflow is result of each partner’s specification and is named global workflow model. Very often business partners have already implemented some kind of locale business logic with security requirements. That means the global workflow describes the interaction of partners abstracting from the internal processing steps and from the internal business logic. In this case, the development of an inter-organizational workflow requires an inside-out proceeding. The interface of the locale business logic is projected onto the global business workflow. Business process modeling is not a topic of this paper, but play important role in modeling of security requirements.

The interface of every partner's node describes the public part of the local business logic, which is accessible to the inter-organizational workflow and must be conforming to a uniform technical, syntactical and semantic specification the partners. The partners should decide on parameter formats, interaction protocols, operation semantics or run-time constraints specification. This kind of partner's requirements is modeled in the interface models and resulted information is typically published in WSDL files and technical models of UDDI registries. The interface model represents a specification of the functional requirements the partner has to implement at its node, as an outside-in proceeding.

From a security perspective, the business process deals with security requirements from the involved partners and makes the secure exchange of messages and documents between different partners. In the view of a partner the local software architecture implements the security portion of the global business workflow and offers a security interface to its partners.

The workflow engine and the Web services in the back-end are wrapped by security components. So-called Policy Enforcement Point (PEP) acts as security gateways. We differentiate between the external and the internal PEP. The external PEP is the single point of entry into the partner's domain. He is in charge of implementing requirements related to message integrity, confidentiality and non-repudiation for all external communication. To this end, it checks the correct signatures and decrypts incoming requests or response. Correspondingly, the gateway signs outgoing requests or responses and encrypts them as specified in the global business workflow. Both, the security gateway and the workflow engine implement the requirements by configuration. The configuration data is generated from the respective models views. After receiving a service request the PEP authenticates the caller with support of the Authentication and Role Mapping, checks the compliance of the incoming message according to signed and encrypted elements and finally queries the Policy Decision Point (PDP) for access rights. After successful completion of these three steps, the PEP forwards the request to the workflow engine which then performs the service orchestration. Optionally, an internal PEP, that merely acts as a role mapping unit may map the caller's global role to some internal role representation required by the back-end applications.

3 MODELING ENVIRONMENT

Our approach to mapping the variant's global security requirements to a customizing modeling tool involves the use of Model-driven development techniques. Different security models of developed customizing modeling tool can be constructed for the automatically check of correctness application designs meet their security requirements. Modeling tools can also be used to generate the customization, composition, packaging, and deployment code to implement security artifacts. To develop such security modeling tool, it is common to build domain-specific modeling languages (DSML) tools on top of a Meta-configurable Modeling Environment. Crucial to creating a viable Meta Modeling Environment is providing a platform that has the appropriate level of abstract for each domain expert. Our Meta Modeling Environment is Generic Eclipse Modeling System (GEMS). The main distinguishing feature of GEMS is an appropriately built presentation meta-model. Generic Eclipse Modeling System contains a universal metamodel-based presentation engine for element property editing, and an advanced project tree engine. The GEMS reuses the basic Eclipse components such as EMF (Eclipse Modeling Framework) and GEF (Graphical Editing Framework), as well as parts of GMF (Graphical Modeling Framework) runtime. GMF utilizes Eclipse EMF and GEF technologies. EMF is used for model management and GEF for graphical user interface. GMF uses a static-mapping-based approach. It defines a set of meta-models: graphical (presentation), tooling and mapping meta-models. Graphical meta-model defines the graphical element types, tooling meta-model defines the palette and menus and the mapping meta-model defines the mapping possibilities between the models. In addition, it uses EMF ECORE Model as the domain meta-model.

The GEMS is created by the Distributed Object Computing (DOC) Group at the Institute for Software Integrated Systems (ISIS) at Vanderbilt University. This Meta-configurable Modeling Environment is enough comfortable and very adaptable for customizing security artifacts via the following capabilities:

- A visual interface supports the creation of domain-specific modeling languages, contains a meta-modeling environment that supports the definition of paradigms, which are type systems that describe the roles and relationships in particular domains.
- The creation of models that are instances of DSML paradigms within the same environment.
- Customization of such environments so that the elements of the modeling language represent the elements of the domain in a much more intuitive manner than is possible via third-generation programming languages.
- A flexible type system that allows inheritance and instantiation of elements of modeling languages.
- An integrated constraint definition and enforcement module that can be used to define rules that must be adhered to by elements of the models built using a particular DSML.
- Facilities to plug-in analysis and synthesis tools that operate on the models.

A key challenge to developing security modeling tools is the formal specification of the security artifacts in DSML. Engineers must create a language to capture both the static and dynamic semantics. The GEMS meta-modeling language allows developers to specify the common types in their system, the variability in instances of the types, the allowed containment compositions of the types, and the allowed connections between instances of types. These capabilities allow developers to apply object analysis to their security artifacts and capture the results in a metamodel. This metamodel can then be interpreted by the GEMS framework to generate DSML(s) for customizing the security artifacts.

By allowing developers to formally define the commonality and the variability in the system and automatically generating security artifacts, Meta Modeling Environment reduces the development effort require for a complete implementation the customization modeling tools, and providing a modeling language intuitive to domain experts that customize the security requirements.

3.1 XACML Policies

The security requirements are configured and expressed in XACML (eXtensible Access Control Mark-up Language), which is an XML based OASIS standard for a policy and access control decision. The main goal of policy generator is to derive low-level and enforceable security policies. These policies are deduced from the business process, which acts as the input scenario. In this scenario, the security requirements should be already identified, and annotated within the business process. The visual programming languages developed with metamodel makes possible to model security requirements in modeling consummation tool. As a part of modeling consummation tool the policy generator generates the access control policies in XACML standard, which defines the machine-interpretable and also machine-enforceable security policies. The generator shall generate follows types of XACML components, Figure 2:

<PolicySet> element combines Policies in a PolicySet. It has a target, a policy-combining algorithm-identifier, a set of policies and obligations.

<Policy> element combines rules in a policy. Therefore, a policy consists of a target element, a rule-combining algorithm identifier, a set of rules and obligations. The target element of policy has the same purpose as target element of rule. The rule-combining algorithm determines rule-combining method applied to the rules.

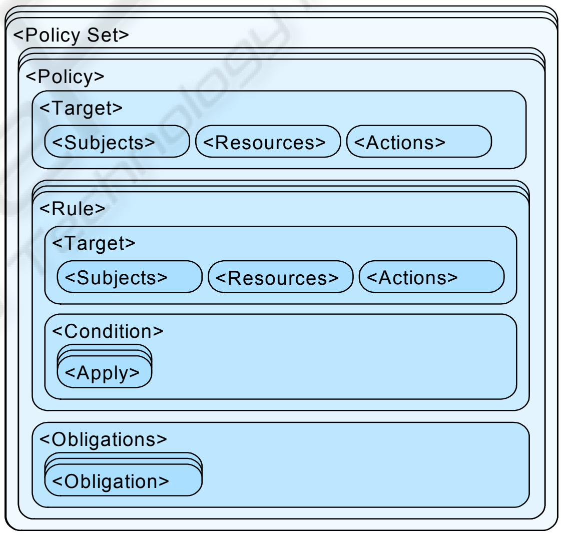

Figure 2: XACML components.

<Rule> element is the most elementary unit of XACML policy. It has the target element, the condition element and the effect element. The target element specifies the environment, on which this rule should apply. The condition element is a Boolean function over subject, resource action and environment attributes. In order to apply the effect of rule on the target, this condition should evaluate to true, otherwise, the rule will return indeterminate. Lastly, the effect element states the decision for this rule: either permit or deny.

<*Target*> element describes the properties of the environment, on which the Rule, Policy or Policy Set should apply. It contains the subject description, resource description, action performed and environment description.

<*Condition*> element represents a Boolean expression that refines the applicability of the rule beyond the predicates implied by its target.

<*Obligations*> element of an XACML <Policy> is a directive to the PEP to perform additional processing following the enforcement of an access control decision. It contains one or more <Obligation> elements and typically references elements in the request context. Processing of <Obligations> elements is application-specific.

4 OUTCOMES AND RESULTS

The first step for building a graphical modeling tool with selected Meta Modeling Environment, called ARCSecu Modeling Tool, is to define a metamodel for specifying syntax of the security artifacts based on the DSML for secure Web services. This metamodel describes the graphical entities, connection types, attributes, and other visual syntax information needed by Meta Modeling Environment. On the Figure 3a and 3b is presented ARCSecu Meta Model with follow artifacts:

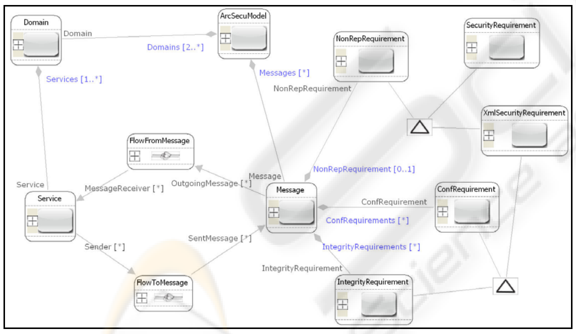

Figure 3a: Metamodel for security requirements.

The first entity (class) is the ArcSecu Meta Model, which is represented as the root entity in our metamodel. Domain, Service and Messages entities describe a collection of abstract operations. Three entity types IntegrityRequirement, ConfRequirement and NonRepRequirement are involved for security artifacts. These security goals are child entities of Message. The Service entity is child entity from Domain.

The ARCSecu Meta Model involves connections between Message and Service. These are defined with FlowFromMessage and FlowToMessage connections in suitable directions. Inheritance relationships are created by adding an Inheritance element to the model and connecting it to the parent type and derived type(s). Any attributes, connections, or containment relationships specified by the parent are inherited by derived types. The presented ARCSecu Meta Model contains entities with two Inheritance relationships. First Inheritance is defined between IntegrityRequirement (BaseClass), ConfRequirement (BaseClass) and XmlSecurityRequirement (DerivedFrom). Second the Inheritance between NonRepRequirement (BaseClass), XmlSecurityRequirement (BaseClass), SecurityRequirement (DerivedFrom).

The classes IntegrityRequirement and ConfRequirement inherit the attributes (NS1, NS2, NS3, NS4, NS5, NS6, NS7, XPath) from XmlSecurityRequirement entity. Service entity carries an attribute (string type) named Operation Name and Message carries also an attribute (Boolean type) named IsResponse.

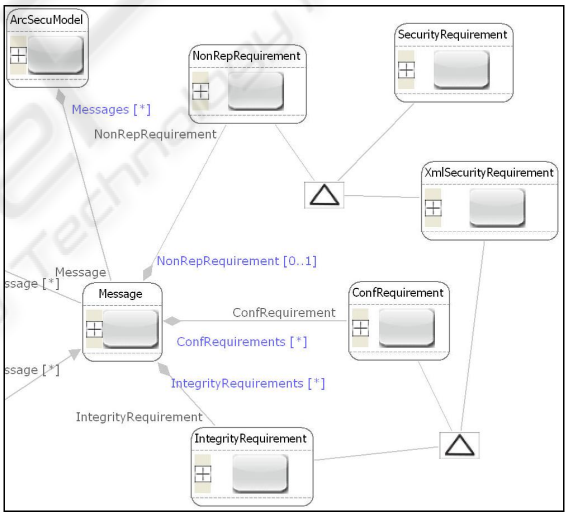

Figure 3b: Metamodel for security requirements – detail.

In step two, a code generator is invoked that produces the EMF, GEF, GMF, XML descriptors, which are needed to create a plug-in for editing the new language based on DSML, described by the metamodel. A graphical design defines custom icons and CSS styles for the DSML elements in the third step. For the fourth step, developers specify the constraints on the model that GEMS uses to ensure that only correct models are built with the model. The constraints generally involve domain information that requires a constraint language to express properly. Finally, domain experts use the customizing modeling tool produced by GEMS to construct models of the domains. Model interpreters (or code generators) are developed to produce software artifacts such as Java code or XML file descriptors.

4.1 Modeling Tool and Generator for the Security Artifacts

The modeling tool as an Eclipse rich client is used to visually modeling the security needs of each communicated domain and it is extended with the XACML policy generator. For each involved domain the generator creates the XACML policy descriptor that needs to be deployed to the policy folder of each domain. To model the communication between two domains one first needs to drag two domain model elements from the palette on the right onto the canvas, Figure 4. The bottom panel of the new domain elements is initially colored red. This indicates that some properties have to be set to make the model element valid. The properties can be set in the properties panel at the bottom of the modeling tool. For each domain a name has to be set that denotes its role in the communication. Next one needs to add services to each domain that communicate with each other by dragging services from the palette on each domain. The services need to have a name and an operation name set to be valid. These actually have to correspond to the real names of the services. Now the communication needs a message that is sent between the two nodes. This is done by dragging a message model element on the canvas.

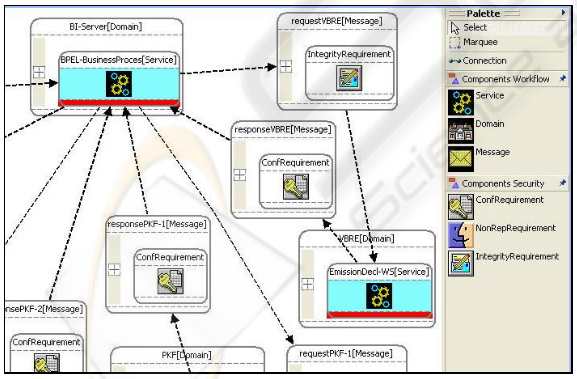

Figure 4: The modeling tool for security requirements.

Then the message has to be associated with the initiating and receiving services by using the connection tool. Finally the security requirements of the communication are configured by dragging security requirements from the palette onto the message. For confidentiality and integrity requirements one has to specify the XPath to the elements on which the requirements shall be enforced and also up to five namespaces. For the non- repudiation requirement no further configuration is needed. The generator can be run by right-clicking on the canvas and selecting ArcSecu Menu->DO IT. For each involved domain the generator creates one folder containing the XACML policies that need to be deployed to the policy folder of each domain.

The security requirements are modeled and translated for exchange of secure information between follows e-government institutions: Municipality (PKF Domain), Environment Ministry Register (eRAS Doamin) and Registry of the Incinerators – annual report about environmental pollution producer (vBRE Domain). The model of this business process is developed with ARCSecu Modeling tool and generator for three security goals: integrity, confidentiality and non-repudiation. Very reduced example of XACML generated policy code is presented on Figure 5.

```
<PolicySet PolicySetId=«PolicySetId» PolicyCombiningAlgId=«PolicyComAlgId»>
 <Target><Subjects>
   <Subject>
    <SubjectMatch MatchId="string-equal">
     <AttributeValue DataType="string">«name1»</AttributeValue>
      <SubjectAttributeDesignator AttributeId="subject-id" DataType="string" />
       </SubjectMatch> </Subject> </Subjects>
  <Resources>
   <Resource>
    <ResourceMatch MatchId="anyURI-equal">
     <AttributeValue DataType="anyURI">«name2»</AttributeValue>
      <ResourceAttributeDesignator AttributeId="resource-id" DataType="anyURI" />
      </ResourceMatch>  </Resource> </Resources>
  <Action><AnyAction /></Action><Target>
 <Policy PolicyId=«PolicyId» RuleCombiningAlgId="deny-overrides">
  <PolicyDefaults>...</PolicyDefaults>
   <Target>
    <Subjects><AnySubject /></Subjects>
    <Resources><AnyResource /></Resources>
    <Actions>
     <Action>
      <ActionMatch MatchId="anyURI-equal">
       <AttributeValue DataType="anyURI">.../RECEIVING</AttributeValue>
        <ActionAttributeDesignator DataType="anyURI" AttributeId="action-id"/>
      </ActionMatch></Action></Actions>
   </Target>
   <Rule RuleId="ReadRule" Effect="Permit">
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="string-equal">
       <Apply FunctionId="string-one-and-only">
        <AttributeSelector RequestContextPath="..." DataType="string" />
       </Apply>
      <AttributeValue DataType="string">«name1»</AttributeValue>
      </Apply>
 ... </Policy>...</PolicySet>
```

Figure 5: Structure of XACML policy.

Client sends the request via web application with the necessary parameters on e-government business service (BPEL service) to get the information about the company's incinerators. BPEL service forwards continuously the request to the secure component of the BPEL server. This prepares the request message in accordance with defined security artifacts in the XACML policy descriptors. This may mean that the message (or parts of the message) is encrypted and/or it be signed and also it holds secured date for authentication and authorization elements. Such secure message is transmitted over the Internet for PKF Domain, which offers the public services with information about the existing companies. The security component of PKF Domain verifies message according to the defined XACML Policy descriptors and the message will be deciphered after successful test and on the PKF Domain servicers redirected. The response of the PKF Web Services is back to the security component and skillfully using the XACML policy again transmitted to BPEL server.

The Business service processes the response over security component to the filter component of Business process, this mean each founded company is checked on registration date in eRAS Domain – the waste handling company register. For filtered companies Business service create a request for vBRE Domain about annual report for environmental pollution from incinerators.

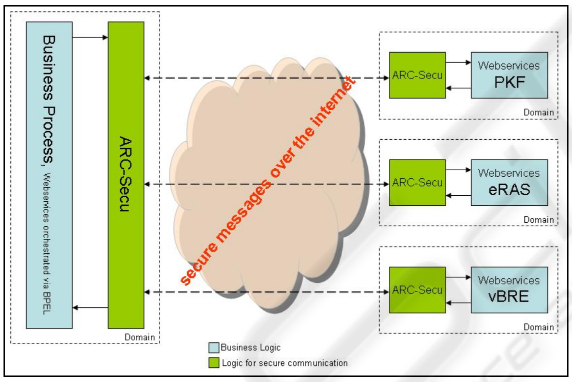

Figure 6: Secure message over internet.

The communication process for other two domains (eRAS and vBRE) follow the similarly security Logic as for PKF domain over predefined XACML policy descriptors, as Figure 6 presented. The verifying of message and integrating security elements in message happen each time by security component of domain after request incoming or before response sending.

5 CONCLUSIONS

We introduce the concept of Model Driven Security for a software development process, which allows for the integration of security requirements through system models. This modeling framework supports the generation of security infrastructures with focuses on business logic as well as on inter-organizational workflow management.

This Meta modeling approach reduces the cost of developing a graphical modeling tool by allowing developers to focus on the key aspects of their tool. The infrastructure to create, edit, and constrain instances of the language is generated automatically and the implementing process makes possible the separation of language development, coding, as well as “look and feel” development.

We have evaluated the Generic Modeling Environment, a configurable modeling and program synthesis tool set, which makes possible the rapid and cost-effective creation of highly customized, domain-specific system. It is highly applicable for modeling XAMCL policy generator related to security artifact for service-oriented architecture. We focus on visually modeling an XACML system to fulfill the XACML profile and automatic generation the security infrastructure in XACML-formatted documents.

This article describes the features available for building graphical modeling tools with Meta Modeling Environment. This Environment focuses on allowing developers to create Eclipse-based modeling tools without writing plug-in descriptors, GMF mapping files, GEF code, or EMF code. Furthermore, Meta Modeling Environment (GEMS) provides facilities to support external specification of visualization details so that they can be managed by graphic designers and other individuals not experienced with complex GUI coding.

The development of a security-critical inter-organizational workflow starts with the analysis and the design of the workflow, followed by a comprehensive analysis of risks and threats, and the ensuing specification of security requirements. The requirements are modeled in a platform-independent way at different levels of abstraction. The project goal is to analyze security issues that may stem from requirements imposed by e-government business processes and in a second step to provide methodologies that translate the abstract security requirements into run-time artifacts for the target architecture through model transformation. Ultimately, the implementation should allow the declaration of the secure message exchange via the internet.

In our proof of concept we used three typically security goals in inter-organizational workflows: integrity, conditionality and non-repudiation. Our aim is on sample way to present our approach for automatic generate security requirements. On the same modeling approach and similar developing of modeling tools is possible to design other security requirements as: Identification, Authentication and Authorization, Trust, Privacy and so on.

Several improvements and extensions need to be addressed in future work. Currently our approach focuses on static design models, which are relatively close to the implementation. It is worth considering whether the efficiency of the development process of secure applications can be improved by annotating models at a higher level of abstraction (e.g. analysis) or by annotating dynamic models. Moreover, some critical questions concerning the development process are still open, e.g. how are roles and permissions identified? Beyond that, the current prototype does not yet demonstrate the platform independence of our concepts. Future work will focus on modeling security requirements and design information using dynamic models. Furthermore, the development process for secure systems starting with the initial analysis up to the complete secure system design will be investigated. In this context, we will examine the possibility of propagating security requirements between analysis and design models and ways to verify the compatibility of requirements and design information given at different levels.

REFERENCES

Xin Jin, Master Thesis, University of Ottawa, Ontario, Canada 2006 *Applying Model Driven Architecture approach to Model Role Based Access Control System*

Taufiq Rochaeli, TUD SEC, Ruben Wolf, Fraunhofer-SIT, *Policy Generator,* February 10, 2006.

Jules White, Douglas Schmidt, Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, USA, *Simplifying the Development of Product-line Customization Tools via Model Driven Development*

Panos Periorellis, Jake Wu, March 2006, *XACML-Role Based Access Control*

Jeremy W. Bryans, John S. Fitzgerald, Panos Periorellis, School of Computing Science, Newcastle University, UK, *A Formal Approach to Dependable Evolution of Access Control Policies in Dynamic Collaborations*

Eduardo Fernández-Medina and Mario Piattini, Alarcos Research Group, Universidad de Castilla-La Mancha, *Towards a Process for Web Services Security*

Yuri Demchenko, Advanced Internet Research Group, University of Amsterdam, *Policy-based Access Control to Data Ser vices in Ser vice-oriented Architecture and Grid*

GEMS EMF Intelligence Tutorial, *http://wiki.eclipse.org/ GEMS_EMF_Intelligence_Tutorial*

GEMS EMF Intelligence Tutorial with Mixed Constraints, *http://wiki.eclipse.org/GEMS_EMF_Intelligence_Tutorial_with_Mixed_Constraints*

GEMS Metamodeling Tutorial, *http://wiki.eclipse.org/ GEMS_Metamodeling_Tutorial*

Jules White, *The Generic Eclipse Modeling System (GEMS)*

Markus Völter, *openArchitectureWare 4.2 Fact Sheet, voelter@acm.org Date:* September 3, 2007

Mirad Zadic, Stockholm, Sweden, 22 - 24 October 2008, *A Meta Model Generator for Implementing Access Control and Security Policies in Distributed Systems based on Model-Driven Architecture, eChallenges e-*2008 Conference & Exhibition

GrTP: Transformation Based Graphical Tool Building Platform, Institute of Mathematics and Computer Science, University of Latvia, *Building Tools by Model Transformations in Eclipse, University of Latvia, Audris Kalnins, Oskars Vilitis1, Edgars Celms1*

OASIS, 2005. eXtensible Access Control Markup Language (XACML) Version 2.0. *http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf*

OASIS, 2005, Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. *http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf*

OASIS, 2005. Core and hierarchical role based access control (RBAC) profile of XACML v2.0. *http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf*

OASIS, 2005. SAML 2.0 profile of XACML v2.0. *http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf*

OASIS, 2005. Web Service Security SAML Token Profile 1.1. *http://www.oasis-open.org/specs/ index.php#wssprofilesv1.0*

OASIS, 2003. XACML profile for Web-services. *http://www.oasis-open.org/committees/download.php/3661/draft-xacml-wspl-04.pdf*

OASIS, 2004. WS-Security 1.1 Core Specification. *http://www.oasis-open.org/committees/download.php/ 16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf*

W3C, 2006. Web Service Policy 1.2-Framework (WS-Policy). *http://www.w3.org/Submission/WS-Policy/*

ORMSC White Paper, *A Proposal for an MDA Foundation Model*

Torsten Lodderstedt, David Basin, and Jürgen Doser Institute for Computer Science, University of Freiburg, Germany, *SecureUML: A UML-Based Modeling Language for Model-Driven Security*