Deriving XACML Policies from Business Process Models

Christian Wolter[1], Andreas Schaad[1], and Christoph Meinel[2]

[1] SAP Research
Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany
{christian.wolter,andreas.schaad}@sap.com
[2] Hasso-Plattner-Institute (HPI) for IT Systems Engineering
University of Potsdam, Germany
meinel@hpi.uni-potsdam.de

Abstract. The Business Process Modeling Notation (BPMN) has become a defacto standard for describing processes in an accessible graphical notation. The eXtensible Access Control Markup Language (XACML) is an OASIS standard to specify and enforce platform independent access control policies.

In this paper we define a mapping between the BPMN and XACML metamodels to provide a model-driven extraction of security policies from a business process model. Specific types of organisational control and compliance policies that can be expressed in a graphical fashion at the business process modeling level can now be transformed into the corresponding task authorizations and access control policies for process-aware information systems.

As a proof of concept, we extract XACML access control policies from a security augmented banking domain business process. We present an XSLT converter that transforms modeled security constraints into XACML policies that can be deployed and enforced in a policy enforcement and decision environment. We discuss the benefits of our modeling approach and outline how XACML can support task-based compliance in business processes.

Keywords: Policy Definition, Integration, Enforcement, Separation of Duties Business Process Modeling, eXtensible Access Control Markup Language.

1 Introduction

The control and audit of activities is a fundamental principle in systems with a high degree of human interaction [1]. In the domain of information systems security and compliance, access control models are used to decide on the ways in which the availability of resources is managed and company assets are protected.

A multitude of access control models and related specification languages have emerged over the last decades. Requirements, such as “access control models must allow high level specification of access rights, thereby better managing the increased complexity [..]” [2], are representative of today’s collaborative, service-integrated, and process-aware information systems. What is evident is the need for business experts to able to define their compliance requirements at a business process level, while the corresponding access control policies need to be specified and enforced at the backend- and service-level of an enterprise information system [3]. In the domain of access control engineering Crampton *et al.* state “[..] that existing approaches to the specification of authorization constraints are unnecessarily complicated” [4]. For instance, this applies to the well defined eXtensible Access Control Markup Language [5]. This XML-based notation is very expressive, but policy definition is cumbersome and tool support is scarce. The manual definition of XACML policies is slow, error prone and may lead to policy inconsistencies and disruption of related business process execution.

We base our work on two key observations with respect to business process driven security administration and compliance management for service-based integrated enterprise environments. First, there is a set of well defined and accessible business process modeling notations used to describe enterprise service orchestration based on Web Services that are widely accepted and standardized. Second, only the security specialists specify security policies for enterprise services on a very technical level to create related security policies instead of the actual stakeholder which is the business process domain expert.

Based on these observations, as an initial step, we had enriched the semantics of the Business Process Modeling Notations (BPMN) in [6] to leverage the specification of access control security policies (e.g., separation of duty) for process-aware information systems onto the level of the process model definition itself. In this paper, we now address the issue of XACML policy specification usability and present a novel approach to derive concrete security policies, from a business process model, transform them into a dedicated policy specification language, such as RBAC-extended XACML policy sets, and deploy these policies in a policy enforcement enriched process execution environment based on Web Services. In essence, in this paper we provide:

- An evaluation of the BPMN and XACML meta-model and description of which entities must be mapped to each other in order to derive XACML security policies from BPMN models.
- A mapping to define a model transformation that extracts policy information from an XML-based business process model description to automate the generation of according XACML policies.
- A proof of concept prototype XSLT transformation script, applied on some example separation of duty constraints taken from a banking process [1].

The rest of this paper is organized as follows. The next section provides some background information about the Business Process Modeling Notation along with our security augmentation for task-based authorization. In Section 3 we discuss the underlying meta-models of BPMN and XACML and identify a mapping between entities of both models in order to define a model transformation to derive XACML policies. In Section 4 we demonstrate the feasibility of our mapping proposal by presenting an XSLT transformation applied to a given example banking process. The process is stored as an XML document and parsed by an XSLT transformer to generate XACML policies. Section 5 presents some related work in the area of policy modeling and model-based policy generation. The last section discusses our current approach and outlines future work, such as policy analysis that goes beyond syntactical validation of policy sets.

2 Background

In this section we provide some short background information about the Business Process Modeling Notation and the eXtensible Access Control Markup Language.

2.1 Business Process Modeling Notation

The Business Process Modeling Notation has emerged as a standard notation for capturing business processes, especially at the level of domain analysis and high-level systems design. The notation inherits and combines elements from a number of other proposed notations for business process modeling, including the XML Process Definition Language (XPDL) [7] and the Activity Diagrams component of the Unified Modeling Notation (UML). Figure 1 depicts an excerpt of the BPMN meta-model. For the sake of simplicity we omitted some elements that are not relevant in the course of this paper. BPMN process models are composed of flow objects such as routing gateways, events, and activity nodes. Activities, commonly referred to as tasks, represent items of work performed by software systems or humans, i.e. human activities. Activities are assigned to pools and lanes expressing organizational institutions, roles and role hierarchies. Routing gateways and events capture the flow of control between activities. Control flow elements connect activity nodes by means of a flow relation in almost arbitrary ways [8]. BPMN supports so called artifacts that enrich the process model by information entities that do not affect the underlying control flow and are a dedicated extension points to add additional information to the model. We provided an extension for security semantics in BPMN by adding an authorization constraint artifact.

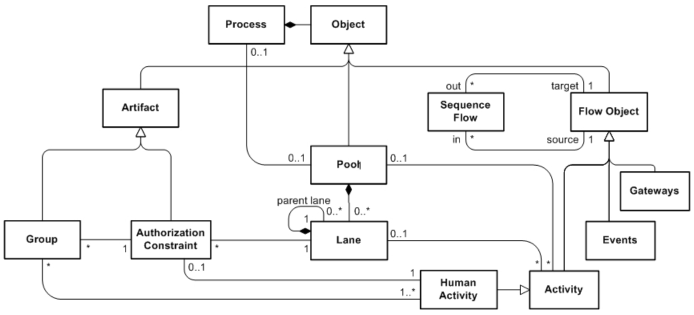


Fig. 1. BPMN Entitiy Diagram Excerpt

2.2 Constrained Business Process

The authorization constraint artifacts we proposed in [6] basically consist of two arguments n_u and m_{th} (cf. Figure 2). The first denotes the number of different users that must perform at least one activity of the set of activities indicated by a group or lane element the constraint is applied to. The latter argument defines the maximum number of activity instances of a given set of activities a single user may perform. This threshold value is necessary to restrict the number of possible task invocations of looped or

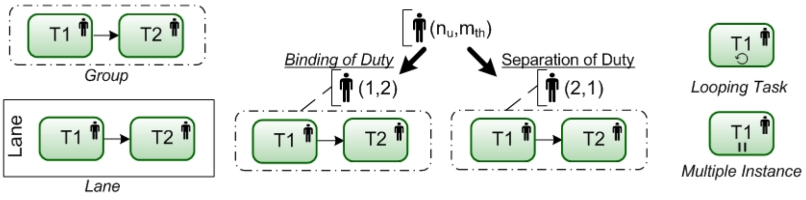
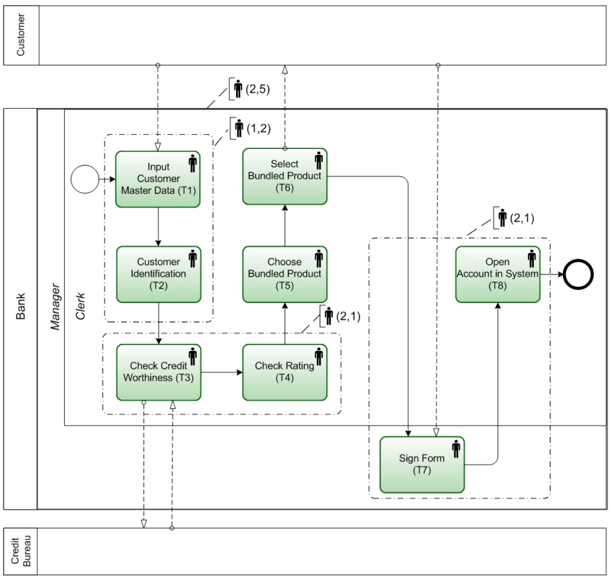


Fig. 2. Modeling of Authorization Constraints [6]



Customer
Bank
Manager
Clerk
Credit Bureau
(2,5)
(1,2)
(2,1)
(2,1)
Input Customer Master Data (T1)
Customer Identification (T2)
Check Credit Worthiness (T3)
Check Rating (T4)
Choose Bundled Product (T5)
Select Bundled Product (T6)
Sign Form (T7)
Open Account in System (T8)

Fig. 3. Example Business Process Model

multiple-instance tasks. Crampton *et al.* defined such constraints as cardinality constraints [4]. The authorization constraint artifact can be assigned to groups, lanes, and repetitive activities.

We applied our notation to a banking process illustrated in Figure 3 and modeled security constraints for this process that are discussed in [1]. The process describes the necessary steps for opening an account for a customer. Therefore, the customer's personal data is acquired. The customer is identified and the customer's credit worthiness is checked by an external institution. Afterwards, one of several product bundles is chosen. A form is printed for the selected bundle that is signed by the customer and the bank.

A role-based authorization constraint for a role *Clerk* is expressed by assigning the set of tasks $\{T1, T2, T3, T4, T5, T6, T8\}$ to the lane labeled *Clerk*. A second role-based authorization constraint for the role *Manager* is expressed by combining the tasks of the role clerk with the task $T7$. The nesting of the lane *Clerk* within the lane *Manager* expresses a role hierarchy. We defined two separation of duty constraints for the pairs of conflicting tasks $\{T3, T4\}$ and $\{T7, T8\}$ by adding the related authorization constraints to both groups. In the same way we expressed a binding of duty constraint for tasks $\{T1, T2\}$. A last operative separation of duty constraint is directly assigned to the lane *Clerk* that restricts the executive power of a single clerk for a process instance to five activities.

2.3 eXtensible Access Control Markup Language

XACML is an OASIS standard that allows the specification of XML-based access control policies, primarily applied to the domain of Web Services. Referring to Figure 4, XACML specifies a request-response protocol and a data flow model between a service requester, a policy enforcement point (PEP), a context handler, and a policy decision point (PDP). Each access request is send to the PEP and forwarded to the context handler. The context handler creates a request context that is unique for each access request. The request context holds a snapshot of the overall system state. For instance audit log information, timestamps, or a subject's organisational context, such as user role. This information is collected from diverse backend systems, e.g. directory services, database systems, or workflow management systems. This request-dependent information is stored as context attributes and is used by the policy decision point to decide whether to grant or deny an access request. A detailed description of the data flow is given in [5].

The meta-model of XACML is shown in Figure 5. The root of all XACML policies is a *policy* or a *policyset* element. *Policysets* may hold one or more *policies* or other *policysets*. Each *policy* contains *rule* elements which are evaluated by the PDP. *Target* elements specify the context a rule applies to or not. A *target* is composed of *subject*, *resource*, and *action* elements. A *subject* element defines a human interacting with the system. A *resource* element defines a protected entity, such as a Web Service, file or process task in the context of a workflow management system. An *action* element defines the operation that is performed on the protected *resource* element. If more than one *rule* applies a rule combining algorithm defines the outcome of the overall decision request.

Condition elements further restrict the overall decision based on the contextual attributes of the access request. A *condition* contains a predicate which evaluates to either *true* or *false*, e.g. a subject's role attribute must be *clerk*. If the *condition* returns *true* the rule's *effect* is returned. The effect may result in a *permit*, *deny*, or *not applicable*

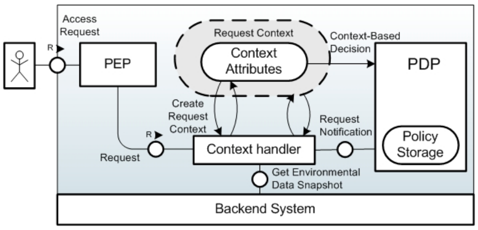


Fig. 4. XACML Overview

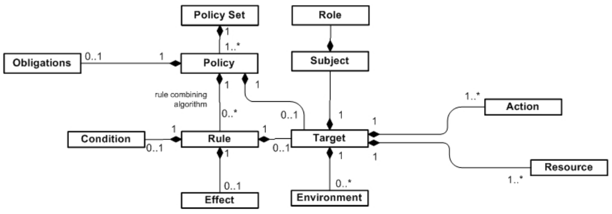


Fig. 5. XACML Entity Diagram Excerpt [5]

statement. An *obligation* is an additional activity (e.g. sending a notification email) that must be performed by the policy enforcement point in case a policy applies to a specific request and rule effect.

3 Mapping of BPMN to XACML Elements

In order to automate the extraction of security policies from process models we discuss in this section the relationship and respective mapping between selected meta-model entities of BPMN and XACML. The dotted lines in Figure 6, indicate the mappings we are discussing in the following:

XACML subject *role* attributes of the policy *target* can be derived from lanes and nested lanes respectively. A lane represents an organisational role, thus the semantic of embedded human activities is interpreted as a role-task assignment.

The BPMN elements *activity* and *human activity* are mapped to XACML *resource* elements as part of the policy *target*. In the context of a workflow management system we consider a process task as a resource. In the domain of process management an activity has several possible states that are related to human interaction, namely *ready*, *activated*, *completed*, or *canceled*. Therefore, for each task we derive three XACML *action* elements for the XACML policy *target*, each related to a state transition that can be performed on a task.

BPMN *group*, *lane*, and *authorization constraint* elements are mapped onto an XACML *condition* element of an XACML policy *rule*. A *condition* describes under which circumstances a rule applies for a matching *target* or not. This supports fine-grained access control and is essential to express separation of duty or binding of duty constraints. In case of separation of duty authorization constraints an XACML *condition* is generated, where the condition must not be met. For instance a specific task must not be performed by the same subject. Binding of duty constraints result in an XACML *condition* element that must be met. For instance, a subject must have executed a previous activity in order to perform the requested activity.

BPMN *authorization constraint* elements also map onto an XACML *obligation* element. *Obligation* elements represent meta-information for an XACML enabled policy enforcement point and contain activities that must be performed by the enforcement point depending on the outcome of the policy evaluation. For instance in case of operational separation and binding of duty constraints potential obligations for a policy enforcement point are extended audit information housekeeping in a backend system to store a subject's activity history information.

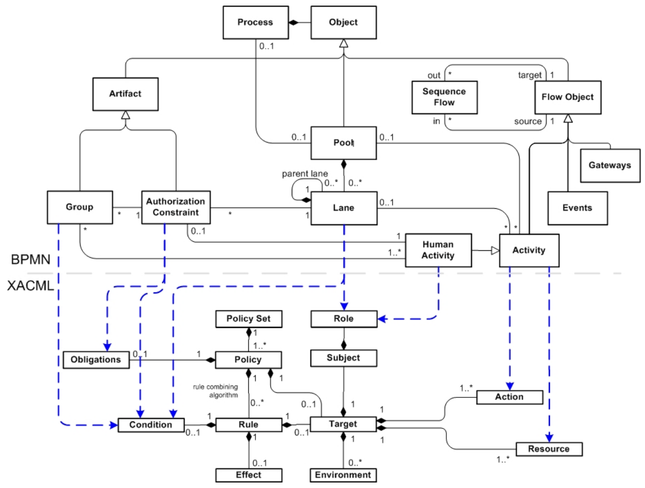
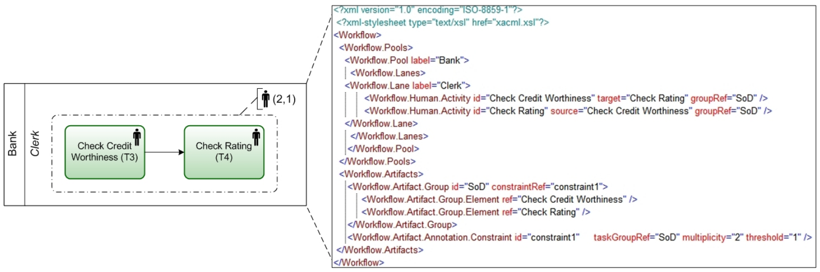


Fig. 6. Mapping of BPMN to XACML

4 Extraction of XACML Elements from BPMN

We developed an XSLT transformation script that automates the generation of XACML policies and stores them in an XML format file that can be read by an XACML based policy decision point. Basically, the script performs the following steps on a given BPMN process source stored in XML-format (cf. Figure 7):



```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <?xml-stylesheet type="text/xsl" href="xacml.xsl"?>
<Workflow>
 <Workflow.Pools>
  <Workflow.Pool label="Bank">
   <Workflow.Lanes>
  <Workflow.Lane label="Clerk">
      <Workflow.Human.Activity id="Check Credit Worthiness" target="Check Rating" groupRef="SoD" />
      <Workflow.Human.Activity id="Check Rating" source="Check Credit Worthiness" groupRef="SoD" />
  </Workflow.Lane>
   </Workflow.Lanes>
  </Workflow.Pool>
 </Workflow.Pools>
 <Workflow.Artifacts>
  <Workflow.Artifact.Group id="SoD" constraintRef="constraint1">
     <Workflow.Artifact.Group.Element ref="Check Credit Worthiness" />
     <Workflow.Artifact.Group.Element ref="Check Rating" />
  </Workflow.Artifact.Group>
  <Workflow.Artifact.Annotation.Constraint id="constraint1"    taskGroupRef="SoD" multiplicity="2" threshold="1" />
 </Workflow.Artifacts>
</Workflow>
```

Fig. 7. BPMN Process Source

1. **Role Engineering**
   The process model is parsed for lane elements. For each lane the script generates a role-based policy set. Each human activity that is assigned to that lane results in a policy and three rules. These rules allow the actions *activate*, *complete*, and *cancel* for each human activity by a subject that holds a role attribute with the corresponding role represented by the lane. If the transformer detects a nested lane a new role-based policy set is created and its contained policies are referenced by the parent lane. According to [5] this allows to express role-hierarchies in terms of role seniority.

2. **Condition Definition**
If the transformer detects an authorization artifact it adds a condition element to the affected policy rule. In the case of separation of duty constraints the logical function of the condition is *NOT*. For binding of duty constraints the logical function *AND* is used. This logical function will be applied to all arguments within the condition element. The authorization constraint references a set of activities either by pointing to a group artifact or a lane element.
3. **Argument Generation**
For each human activity defined in a referenced task group or lane element an attribute function is applied to the condition element. This element takes a subject identifier, a process identifier, a set of task identifier (i.e. argument bag), and an optional threshold value as arguments. We defined an abstract *check:history* function for the XACML context handler able to query audit log information from workflow systems. Depending on the utilized system this function must be adapted. The *check:history* function returns *true* or *false* if the subject has performed any of the tasks referenced by the set of tasks in the argument bag.

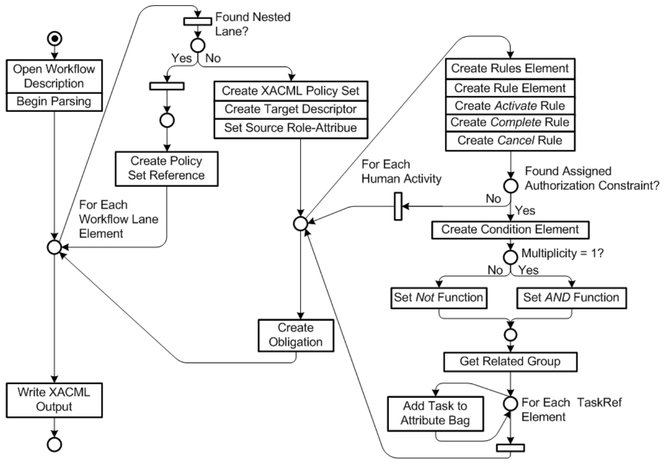
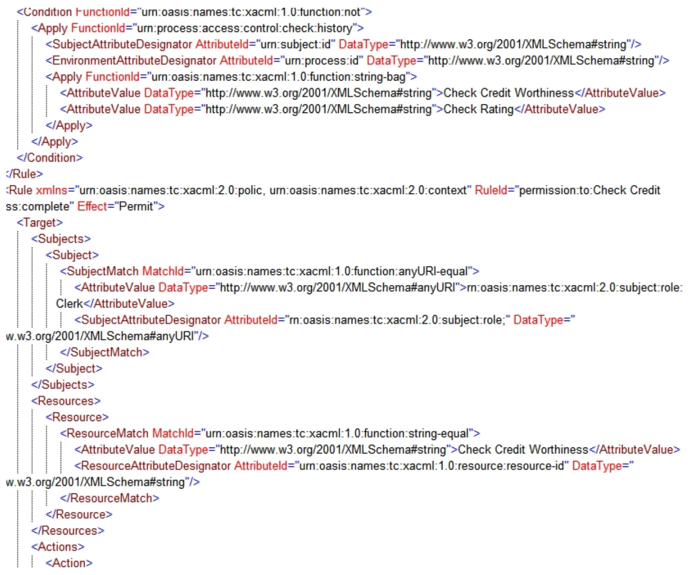


Fig. 8. Policy Generation as a Petri-Net

The described steps are implemented by an XSLT script. The overall algorithm is shown in Figure 8 as a Petri-Net. This script extracts information from a BPMN source process, such as shown in Figure 7, and stores them as XACML policies. Please note, due to the length of the overall transformation we can only provide a small script extract in this paper. Parts of the resulting XACML policy set that is generated based on our example (cf. Figure 7) is depicted in Figure 9. The extract shows a separation of duty condition between the two tasks *Check Credit Worthiness* and *Check Rating* as well as a simple task-role assignment of the task *Check Credit Worthiness* to the role *Clerk*. Regarding the syntactical complexity of a simple separation of duty policy between two tasks in XACML, it is comprehensible why XACML lacks a wide spread deployment in



```
  <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
    <Apply FunctionId="urn:process:access:control:check:history">
      <SubjectAttributeDesignator AttributeId="urn:subject:id" DataType="http://www.w3.org/2001/XMLSchema#string"/>
      <EnvironmentAttributeDesignator AttributeId="urn:process:id" DataType="http://www.w3.org/2001/XMLSchema#string"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Check Credit Worthiness</AttributeValue>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Check Rating</AttributeValue>
      </Apply>
    </Apply>
  </Condition>
:/Rule>
:Rule xmlns="urn:oasis:names:tc:xacml:2.0:polic, urn:oasis:names:tc:xacml:2.0:context" RuleId="permission:to:Check Credit
ss:complete" Effect="Permit">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">rn:oasis:names:tc:xacml:2.0:subject:role:
       Clerk</AttributeValue>
          <SubjectAttributeDesignator AttributeId="rn:oasis:names:tc:xacml:2.0:subject:role;" DataType="
w.w3.org/2001/XMLSchema#anyURI"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Check Credit Worthiness</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="
w.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
```

Fig. 9. Resulting XACML Policy Set Extract

enterprise environments and how our model transformation approach to automatically derive security policies from business process models may mitigate this situation.

While this description is tailored for the BPMN meta-model, our approach is general in nature and can be applied to other process modeling notations, for instance XPDL [7] or jPDL [9].

5 Related Work

Our integration approach of BPMN and XACML on the meta-model level is related to some areas of model-driven security and general modeling of authorizations in workflow systems.

In the domain of UML-based security modeling several approaches exist, such as UMLSec [10], SecureUML [11], or a modeling methodology presented by Dobmeier and Pernul in [12]. These approaches allow to model fine-grained access control on the application level for accessing services in open systems, such as Web Services, by terms of generating code fragments from UML models, but they do not address access control in the context of workflow management systems. In SECTET [13] a novel approach for the specification of model-driven access rights for service-oriented architectures is presented. In SECTET security policies are expressed in a predictive language and are translated into platform independent XACML permissions interpreted by a security gateway. A similar approach is taken in [14] by deriving XML-based security policies from policies described in natural language. They created a set of grammars which execute on a shallow parser that are designed to identify rule elements in natural language policies. In the domain of graphical security policy specification Hoagland *et al.* developed LaSCO [15] a graph-based specification language for security policies. LaSCO policies are specified as expressions in logic and as directed graphs, giving a visual view of the policy. A LaSCO specification can be automatically translated into executable code that checks an invocation of a Java application with respect to a policy. In [16], Neumann *et al.* discussed a graphical role engineering and administration tool to define authorization constraints that can be enforced as part of RBAC context constraints.

In [17] and [18] various workflow meta models are analysed in order to evaluate the capabilities to model a workflow as a set of relating roles. Roles are defined in term of goals, qualifications, obligations, permissions, protocols, etc. Nevertheless, they did not address the problem of task-based authorization and their enforcement in a process-aware environment. In [19], Atluri *et al.* define role allocation constraints in the context of workflows and then assign one or more users to each role. They are able to express security policies, such as separation of duties, as constraints on users and roles and developed a constraint consistency analysis algorithms, but provide no translation to actually enforce their policies in an information system. In the area of model transformation for business processes Strembeck *et al.* [20] presented a transformation algorithm to extract RBAC policies from process models expressed in BPEL. They present an approach to integrate Role-Based Access Control (RBAC) and BPEL on the meta-model level and automate steps of the role engineering process. Transforming process models into a machine executable notation is also discussed in [21] by Aalst *et al.* They defined a mapping to convert the control-flow structure of a BPMN source model into BPEL, a language supported by several process execution platforms. The workflow meta-model described in [22] by Leymann and Roller also contained access control elements, but omitted a detailed discussion and extraction of enforceable security policies.

6 Conclusion

In this paper we presented an approach for the automated derivation of authorization constraints from BPMN model annotations to enforceable XACML policies. This was based on an analysis of the two corresponding meta-models and the definition of an appropriate transformation algorithm. Our work is motivated by two main observations.

First, the definition of XACML policies is overly complex, cumbersome, and time consuming. This may result in syntactical and semantical errors. The automated generation of XACML policies from a modeling notation that is more accessible by humans than directly editing XACML policies will speed up the whole policy engineering process. It allows for the direct definition of policies in the context of the underlying business process, avoiding potential inconsistencies between defined security policies and process models based on model changes.

Second, the usability and benefits of our compliance extensions for the Business Process Modeling Notation are strengthened by the mapping to XACML policies that can be directly enforced in an enterprise environment based on Web-Service orchestration and demonstrate how the complexity of XACML-based policy administration can be reduced by defining them on a more abstract level. A reduced complexity of XACML policy administration may lead to a wider roll-out of the XACML standard in enterprise information systems and overall increased general acceptance.

In future work we plan to integrate an XACML-based policy enforcement and decision point into a process execution engine that works in conjunction with an user management service. As mentioned in Section 4, we will implement the XACML context handler extension *check:history* capable of accessing audit logs and activity specific history information stored in workflow management systems along with a new XACML obligation method used for policy-based auditing. In conjunction, both will support task-based compliance for business critical processes that goes beyond traditional role-based security and access control lists. Another interesting aspect would be to apply a consistency checking algorithm, such as proposed in [4], to avoid the creation of contradicting policies either on the XACML model or the BPMN model itself. From a model transformation perspective, we also think about bi-directional model transformation approaches to enable the import of existing XACML policies into a given process model in order to support system migration scenarios.

References

1. Schaad, A., Lotz, V., Sohr, K.: A Model-checking Approach to Analysing Organisational Controls in a Loan Origination Process. In: SACMAT 2006. Proceedings of the eleventh ACM symposium on Access control models and technologies (2006)
2. Tolone, W., Ahn, G.-J., Pai, T., Hong, S.-P.: Access control in collaborative systems. ACM Comput. Surv. 37(1), 29–41 (2005)
3. Schreiter, T., Laures, G.: A Business Process-centered Approach for Modeling Enterprise Architectures. In: Proceedings of Methoden, Konzepte und Technologien für die Entwicklung von dienstebasierten Informationssystemen (EMISA) (2006)
4. Tan, K., Crampton, J., Gunter, C.: The consistency of task-based authorization constraints in workflow systems. In: CSFW 2004. Proceedings of the 17th IEEE workshop on Computer Security Foundations (2004)
5. Anderson, A.: Core and hierarchical role based access control (RBAC) profile of XACML v2.0. OASIS Standard (2005)
6. Wolter, C., Schaad, A.: Modeling of Authorization Constraints in BPMN. In: BPM 2007. Proceedings of the 5th International Conference on Business Process Management (2007)
7. The Workflow Management Coalition.: Process Definition Interface – XML Process Definition Language (2005), `http://www.wfmc.org`
8. Dijkman, R.M., Dumas, M., Ouyang, C.: Formal Semantics and Automated Analysis of BPMN Process Models. In: ePrints Archive (2006)
9. Red Hat Middleware.: JBoss jBPM 2.0 jPdl Reference Manual (2007), `http://www.jboss.com/products/jbpm/docs/jpdl`
10. Jürjens, J.: UMLsec: Extending UML for Secure Systems Development. In: UML 2002. Proceedings of the 5th International Conference on The Unified Modeling Language, pp. 412–425 (2002)
11. Basin, D., Doser, J., Lodderstedt, T.: Model Driven Security for Process-Oriented Systems. In: SACMAT 2003. Proceedings of the eighth ACM symposium on Access control models and technologies, pp. 100–109 (2003)
12. Dobmeier, W., Pernuk, G.: Modellierung von Zugiffsrichtlinien für offene Systeme. In: Tagungsband Fachgruppentreffen Entwicklungsmethoden für Informationssysteme und deren Anwendung (EMISA 2006) (2006)

13. Alam, M., Breu, R., Hafner, M.: Modeling permissions in a (u/x)ml world. In: ARES 2006. Proceedings of the First International Conference on Availability, Reliability and Security, Washington, DC, USA, pp. 685–692. IEEE Computer Society Press, Los Alamitos (2006)
14. Brodie, C.A., Karat, C.-M., Karat, J.: An empirical study of natural language parsing of privacy policy rules using the sparcle policy workbench. In: SOUPS 2006. Proceedings of the second symposium on Usable privacy and security, pp. 8–19. ACM Press, New York (2006)
15. Hoagland, J.A., Pandey, R., Levitt, K.N.: Security Policy Specification Using a Graphical Approach. In Technical report CSE-98-3. The University of California, Davis Department of Computer Science (1998)
16. Neumann, G., Strembeck, M.: An approach to engineer and enforce context constraints in an rbac environment. In: Proc. of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT) (2003)
17. Muehlen, M.z.: Evaluation of workflow management systems using meta models. In: HICSS 1999. Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences, Washington, DC, USA, vol. 5, p. 5060. IEEE Computer Society Press, Los Alamitos (1999)
18. Yu, L., Schmid, B.: A conceptual framework for agent-oriented and role-based work ow modeling. In: Proc. of the 1st Int. Workshop on Agent-Oriented Information Systems (1999)
19. Bertino, E., Ferrari, E., Atluri, V.: The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. ACM Transactions on Information and System Security 2, 65–104 (1999)
20. Mendling, J., Strembeck, M., Stermsek, G., Neumann, G.: An approach to extract rbac models from bpel4ws processes. In: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), Modena, Italy (June 2004)
21. Ouyang, C., van der Aalst, W.M.P., Marlon, D., ter Hofstede, Arthur, H.M.: Translating BPMN to BPEL. In: BPM Center Report BPM-06-02 (2006)
22. Leymann, F., Roller, D.: Production Workflow: Concepts and Techniques. Prentice Hall PTR, Upper Saddle River (2000)
23. Object Management Group.: Business Process Modeling Notation Specification (2006), www.bpmn.org