An MDA-Based Environment for Generating Access Control Policies

Heiko Klarl[1,2], Florian Marmé[1,3], Christian Wolff[2], Christian Emig[1,3], and Sebastian Abeck[3]

[1] iC Consult GmbH, Keltenring 14, 82041 Oberhaching, Germany
[2] Media Computing, University of Regensburg, Germany
[3] Cooperation & Management, University of Karlsruhe (TH), Germany

Abstract. Identity management and access control are essential in the enterprise IT landscape in order to control access to applications and to fulfil laws or regulations. The global competition of enterprises leads to short development cycles and fast changes of IT applications, which requires also an error-free and quick adaption of its security. The model-driven development of access control policies promises to cope with this situation. This work introduces an mda-based environment for generating access control policies. A comprehensive overview is given on the organisational aspects, describing details of roles, artefacts and tools involved. On this basis the four phases of a model-driven development process for access control policies and their organisational aspects are presented.

Keywords: Model-driven Security, Access Control Policies, Identity Management, Model-driven Development Process

1 Introduction

In modern enterprises business processes are extensively supported by IT systems. In order to cope with competitors and to react to demands of the global markets, adoptions of new business requirements have to be handled quickly. This leads to concerns for the companies' IT systems, business processes and their security architecture [1]. On the one hand, functional business requirements have to be implemented very fast. In order to fulfil those demands, *business process management* (BPM) [2] and *model-driven approaches* [3,4,5] have developed over the last years and have proven their benefits within software engineering. On the other hand, securing IT systems, especially the access control aspect, is still heavily based on documents without a formal and standardised structure, at the same time exposing a gap between the business and the IT perspective: In addition, there is a lack of adequate tools supporting the business side in expressing their security needs. Thus, the IT department, which has to implement the business's requirements, has to adjust both views. Typically, this is based on a rather unstructured communication process with many phases of human interaction which is error-prone, in the worst case resulting in an insufficiently secured IT landscape. We propose a model-driven development process of access

control policies as an improvement of the current situation: Modelling access control policies starts from a business perspective, when early business-focused models of the business process are enriched with access control statements. After this starting point a continuous refinement and transformation phase begins, leading to access control policies for a specific security product. In this paper, we give a comprehensive overview of this model-driven development process for access control policies. The different phases of the development process are described in detail and a mapping of roles, artefacts and tools to the corresponding phases is shown. The paper is organised as follows: In Section 2 the background and related work are analysed. Organisational aspects like roles, artefacts and tools are presented in Section 3, while the four phases of the development process and its relation to organisational aspects are shown in Section 4. The paper concludes with a summary and an outlook on future work in Section 5.

2 Background and Related Work

Before we discuss related work, a short overview of the work on which this paper is built upon is given, touching the fields of identity management, access control and model-driven security in the context of service-oriented architecture and business process modelling. In [6,7] *Web Services Access Control Markup Language* (WSACML) has been introduced, an access control meta-model for service-oriented architecture. WSACML describes policies at the *platform independent* level, which means that they are specified with detailed technical resource descriptions while *platform specific* details and notations for specific security products are not covered. Based on this work we have developed the WSACML2PSSP transformation tool [7], which allows the generation of platform specific security policies out of WSACML policies for commercial products e.g. *CA SiteMinder*. In order to cover access control from a business point of view as well, we describe how business processes can be enriched with security artefacts to express access control requirements [8]. In a prototypical implementation, the *UML activity diagram* has been extended with a UML2 profile for identity management. To attach access control statements to the business process, this extension introduces *DraftedPermission* and *Policy* elements. We have also applied this extension concept to the *Business Process Modeling Notation* (BPMN), adapting a BPMN modelling tool to use our extensions within graphical modelling. As the business process model is a computation independent model from a business point of view, we showed in [9] how this model could be transformed to a platform independent model, fitting the WSACML meta-model. In a broader perspective, these developments allow for the generation of platform specific security policies out of computation independent secured business process models utilising the WSACML meta-model at the platform independent level. While these studies strongly focus on specific aspects of access control, secure business process modelling, and model-driven techniques, a more general view describing the development process of access control policies as a whole is still missing. Especially the organisation of the development process

and its environment like roles, artefacts of the single phases and the utilisation of tools is still not adequately shown in a comprehensive view.

As far as related work is concerned, the management of model-driven security is taken into focus in [10]. The authors make use of model-driven security for service-oriented architectures in order to cope with the steady change of applications and with the error-prone administration of security polices, introducing the *OpenPMF* framework. Some technical details of the framework are shown, but no concept of a model-driven development process for security policies is presented. In addition, there is no discussion of organisational aspects which are necessary to generate security policies in a model-driven way. Rees et al. introduce *PFIRES*, a policy framework for interpreting risk in e-business security in [11]. This framework is dedicated to high level policy management and it describes a policy life-cycle which is aligned with the software development process. The paper gives an overview on policy management and its phases from a global perspective, but it does not address a specific implementation scenario, where access control policies are developed to secure e.g. a business process. In [12], Rodríguez et al. present *M-BPSec*, a method for security requirements elicitation for business processes. Their four-phase method is decoupled from the software development process of functional components. As the method ends with the transformation of a *secure business process* into *analysis-classes* and use case diagrams, the creation of platform specific security policies is beyond the scope of this approach and there is no description of artefacts, tools, and roles covering organisational aspects. In [13] Nagaratnam et al. describe a method for business-driven application security. They introduce different roles involved in the software development process, as well as the tools used and show how an application may be brought from the analysis phase to operation: The integration of security requirements starts with the modelling of *security intentions* within business process models, which are understandable from a business perspective. Specific security requirements may be specified later on, e.g. in a UML class diagram. In contrast to our approach, the authors do not propagate a full model-driven development process for access control policies. Limiting the business view to modelling *security intentions* neglects the fact that many security requirements are driven by business needs which means that the business should have knowledge of them. The suggested use of tools is focused on a single vendor and does not comprise tools that specifically support the development of security policies.

3 Organisational Aspects – Roles, Tools and Artefacts

The model-driven development of access control policies is not just a simple sequence of single steps which have to be processed, it is also a complex process which involves different *roles*, *tools* and *artefacts*. Roles organise employees according to their responsibility in the enterprise. A variety of tools is utilised to support the development process. Artefacts of different kind are used as input data or are created as output within single steps of the development process. This

section gives an overview on these aspects before the model-driven development process for access control policies is introduced in the following section.

3.1 Roles Involved in the Development Process

The following roles in the development process are discussed below:

- Process Owner
- Business Analyst
- Security Architect
- Security Developer
- Security Administrator

The *process owner* possesses the business process and is often represented by an organisational unit, not necessarily by a single person. From a business point of view he is in charge of the business process. His interest is to support his business by utilising the business process, therefore he is the source for improvement requests to satisfy business requirements. The process owner has a deep knowledge of all business-related aspects regarding the business process and he knows which laws or regulations have to be followed. He is able to clarify who is allowed to use the business process, a precondition for formulating *access control requirements*.

The *business analyst* is the interface between business-related and IT departments. He is experienced in *business process modelling* (BPM) and knows which business requirements can be successfully realised with the help of the IT. A deep understanding of the business domain with its requirements and characteristics enables him to link business and IT views of the problems involved.

The *security architect* has expertise in IT security architectures, especially in identity management and the modelling of access control policies. He is able to understand the non-formalised *DraftedPermission* elements and resolves them into formalised *Policy* elements. After that he will check whether all *Policy* elements are attached to the business process in a reasonable way and, if necessary, he will fix misconceptions and modelling errors.

The *security developer* is a security expert focusing on the implementation of platform specific access control policies. The security developer is responsible for generating WSACML policies out of *Policy* elements with the help of the PE2WSACML transformation tool (see sec. 3.3 for the discussion of tools involved in our approach). He has to resolve warnings and errors which are thrown during the transformation process and has to link the generated WSACML policies to service operations within the company's service-oriented architecture. In order to use the access control policies in a commercial product, the security developer creates product specific access control policies with the help of the WSACML2PSSP transformation tool.

The *security administrator* is in charge of the security infrastructure's operation, i.e. he is monitoring the systems, installing product updates, deploying and updating access control policies as well as solving problems with the security

infrastructure. He has to ensure that new deployments of policies will not cause problems in the production environment and that the import of these updates will be successfully distributed in the company's security infrastructure.

3.2 Artefacts in the Development Process

The *DraftedPermission* element contains access control statements in a non-formalised and possibly incomplete manner. It enables the business site to describe their desired access control behaviour in prose which has to be refactored into *Policy* elements later. In order to use the *DraftedPermission* element within business process modelling, we have extended the *Unified Modeling Language* (UML) activity diagram and the *Business Process Modeling Notation* (BPMN). The *DraftedPermission* element has been introduced in [8], and further information can be found in [9].

The *Policy* element contains *formalised* access control statements. Attributes for describing their compliance and security status ease the handling of them. *Policy* elements are reusable, i.e. it is intended to reuse formerly defined policies when access control requirements reappear in another business scenario. The *Policy* element has been proposed in [8,9].

WSACML policies are based on an access control meta-model for web service-oriented architectures which has been introduced in [6]. WSACML describes access control policies at a *platform independent level*, i.e. they are specified with detailed technical resource descriptions: For example, the service operation to be protected is specified, but *platform specific details* on security issues and notations for security products are not covered.

Platform specific security policies are policies for a certain product, e.g. for a commercial off the shelf product. They describe access control requirements in the product's vocabulary. Platform specific security policies can be in a proprietary or standardised format like XACML; but as of 2009 most commercial products are not fully standard compliant.

The *security-enriched business process model* is described in an extended BPMN notation, supporting the annotation of security relevant information. The security-enriched business process model has its position at the beginning of the development process for access control policies which means that some work on securing the process has already been completed while other steps are still missing. The model has (formal) *Policy* elements attached to it, but also *DraftedPermission* elements containing security requirements written in prose which have to be resolved in a later phase. The business process model is enriched with access control statements but needs manual refinement for completely supporting a model-driven generation of access control policies.

The *secured business process model* in BPMN notation contains formalised *Policy* elements only. It is created out of the security-enriched business process model after all *DraftedPermission* elements are resolved into *Policy* elements. The business process model itself is the *platform independent* starting point for a model-driven development process resulting in the generation of platform specific access control policies for a commercial product.

3.3 Tools and Repositories Supporting the Development Process

Based on the process model sketched above we have developed a number of modelling and transformation tools which support the mda-based creation of access control policies. At the same time, relevant artefacts are stored in repositories which are used in the policy transformation process. Both, tools and repositories are described below.

BPMN modelling tools support the modelling of business processes. The tools provide all BPMN elements, check whether they are assembled in a correct way or support the graphical arrangement of model elements. In order to use a BPMN modelling tool for modelling access control requirements within the business process, the tool has to support an extension mechanism to include new elements like *DraftedPermission* or *Policy*.

A *business object repository* stores enterprise-wide entities like *order*, *customer* or *contract* which are used to describe input or output data for different activities. Each department within the enterprise has a different view on those entities. For example, the sales department defines a customer as a person who has already ordered a product, whereas the marketing department defines a customer as a person that should be convinced to place an order. As potentially contradictory views can cause difficulties in the modelling of business processes, the representation of the business object should be stored in a *business object repository* which includes all aspects of the entity within the enterprise. We developed it as a directory which supports the search for existing business objects and its attributes to utilise and interpret them in business process modelling.

The *vocabulary repository* was designed in order to support the creation of access control policies. The formal description of policies ensures syntactic correctness, but the wrong usage of attributes, e.g. the usage of non-existent attributes will lead to semantically invalid policies. To overcome this weakness, the vocabulary repository contains all attributes which may be used. These are for example *business roles*, *environment attributes*, names of existing *services* and *service operations*. When a policy has to be created, all attributes used can be looked up in the vocabulary repository. In addition, during transformations of policies all attributes are checked as well.

The *policy repository* contains formal security *Policy* elements which are already used in modelled business processes. The computation independent policies are stored in an XML representation we have proposed in [9]. We designed the policy repository to support the reuse of policies within the business process model in case of identical security requirements. As some access control policies reoccur frequently in different business processes, the reuse of existing policies not only saves work but also contributes to increased security as the policies stored in the policy repository have been reviewed by security experts beforehand. The policy repository is searchable so that policies fitting the user's requirements can be looked up. Meta-data fields describe the policy's content as well as security and compliance aspects.

The *WSACML policy repository* stores platform independent WSACML policies. For every policy to be stored in the repository, the tool checks whether the

policy should be created or whether existing one should be updated. WSACML policies are platform independent but already linked to service operations, for example. The WSACML repository gives an overall view on access control statements for a certain service operation, as only WSACML policies are linked to entities of the IT systems architecture.

The *PE2WSACML transformation tool* extracts *Policy elements* (PE) out of a *secured business process model* and transforms them into *WSACML policies*. During the transformation process it utilises the *vocabulary repository* in order to check whether all attributes used within the policies exist and are used in a correct way. Failures that are detected during this check are displayed as warnings and error messages and have to be corrected in the *secured business process model* afterwards, before rerunning the transformation.

Platform independent WSACML policies have to be transformed to platform specific policies by the *WSACML2PSSP transformation tool* in order to be used by a commercial security product. This tool handles the meta-model of WSACML as well as the meta-models of all supported security products. Transformation rules between WSACML and the target product describe in which way the content of WSACML policies is mapped to the elements of the product's platform specific policies. This approach allows the generation of arbitrary platform specific policies as only the mapping rules to the new meta-model have to be reengineered. A specific example for this mapping has been given for *CA SiteMinder* in [7].

The *policy import tool* is a generic name for the mechanism to import platform specific security policies in a certain security product. Depending on the product the import of new or updated policies works in different ways. Some tools provide an interface allowing the import of policies by loading them into the product. Other products need a customised policy import tool, developed with the product's specific *application programming interface* (API). A policy import tool in the context of *CA SiteMinder* is used in [7].

Fig. 1 shows the relationship between transformation steps, repositories and modelling and transformation tools.

4 Model-driven Development of Access Control Policies

In analogy with the software development process, the model-driven development process for access control policies has four phases. It starts with the modelling of the business process and its access control requirements and ends with the deployment of platform specific security policies within the IT security architecture of the enterprise. Focusing on the model-driven development of access control policies, functional aspects within the different phases are not shown in detail below. For every phase within the development process we describe specific activities and the roles performing them, the artefacts required and produced, like documents or models, and the tools employed. The roles are examples for the job function the persons carry. They are heavily dependant on the organisational

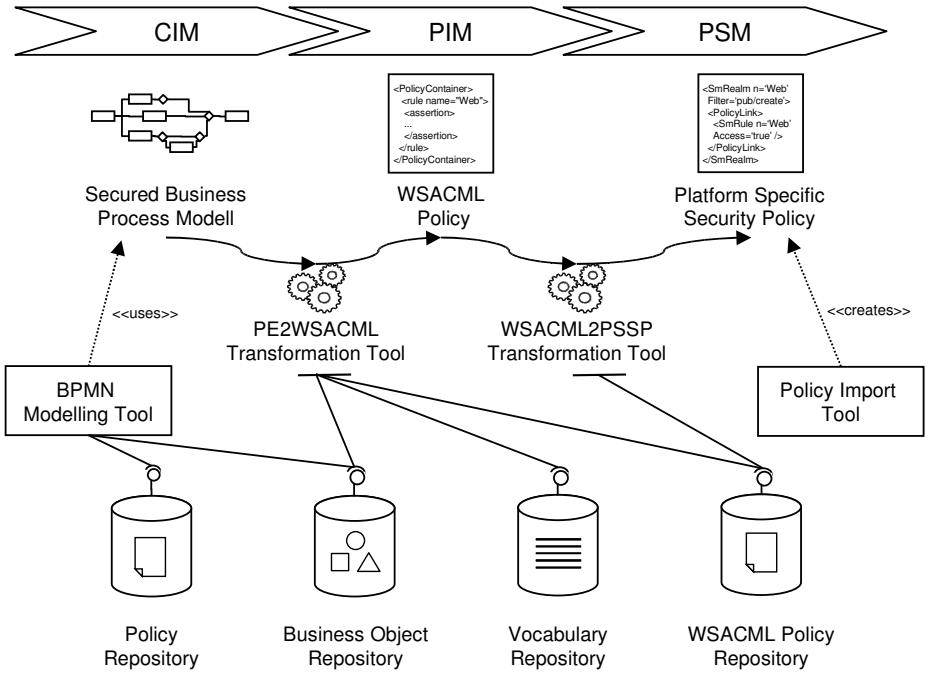

Fig. 1. Architecture Overview

structure of the enterprise. Table 1 gives an overview of roles, artefacts, and tools involved in our approach to model-driven development of access control policies.

4.1 The Analysis Phase

The analysis phase begins with the definition of details regarding the business process. This is done by the *process owners* in collaboration with a *business analyst*. As soon as all facts and requirements are collected, the business process is modelled in a computation independent way, e.g. in the *Business Process Modeling Notation* (BPMN). In order to ease the handling of business objects during modelling, the *business object repository* can be utilised. As access control requirements are driven by business needs, the team should try to collect them in this early stage of the development process and attach them to the business process model. This can be done either in the form of *DraftedPermission* elements (see sec. 3.2 above) or by reusing *Policy* elements from the *policy repository*. Utilising the process owners' knowledge on access control restrictions within their domain and coupling them with the business process model will ensure that all security requirements are covered during the development process. The result of the analysis phase is a *security-enriched business process model* including formal as well as non-formal access control requirements specified from a business point of view.

Table 1. Roles, Tools and Artefacts within the Development Process

		Phases			
		Ana.	**Des.**	**Impl.**	**Depl.**
Roles	Process Owners	x			
	Business Analyst	x			
	Security Architect		x		
	Security Developer		x	x	
	Security Administrator				x
Artefacts	DraftedPermission Elements	x	x		
	Policy Elements	x	x		
	WSACML Policies		x	x	
	Platform Specific Security Policies			x	x
	Security-enriched Business Process Model	x	x		
	Secured Business Process Model		x		
Tools	BPMN Modelling Tool	x	x		
	Business Object Repository	x	x		
	Policy Repository	x	x		
	Vocabulary Repository		x		
	WSACML Policy Repository		x		
	PE2WSACML Transformation Tool		x		
	WSACML2PSSP Transformation Tool			x	
	Policy Import Tool				x

4.2 The Design Phase

In the design phase, the output of the analysis phase (i.e., a *computation independent model* (CIM)) is the basis for the development of the *platform independent model* (PIM). Tasks defined in the business process model have to be mapped to existing service operations or to service operations that have to be created. As the functional development process goes beyond the scope of this paper, the necessary steps will not be described; for further information on this procedure we refer to [14]. From a security point of view *DraftedPermission* elements within the business process model have to be transformed to formalised *Policy* elements. This will be done by a *security architect*. At this point all *DraftedPermission* elements have to be resolved and the business process is designed from an IT perspective. The resulting artefact is the *secured business process model* and all *Policy* elements are stored in the *policy repository*. As a next step, the generation of platform independent *WSACML policies* can be done by the *security developer* using the *PE2WSACML transformation tool*, which reads policies from the *secured business process model* and transforms them into *WSACML policies* which are stored in the *WSACML policy repository*, at the same time unifying all *WSACML policies* linked to the same service operation. During the transformation process the attributes used within the policies are validated using the *vocabulary repository* and the *business object repository* to ensure that only valid attributes and values are used. Warnings and errors thrown by the transformation tool have to be corrected by the *security developer*. As a last step

the manual mapping of the business process task names to service operations has to be done which assigns the *WSACML policy* to a specific service operation.

4.3 The Implementation Phase

As companies employ security products from a broad number of vendors, *platform specific security policies* e.g. for *CA SiteMinder* or *IBM Tivoli Access Manager* must be generated. The *WSACML2PSSP transformation tool* operated by the *security developer* uses *WSACML policies* as input and generates platform specific security policies. The transformation runs automatically and there is no need for manual rework or adaption. All plausibility checks on the attributes used within the policies have already been done in the design phase. The *WSACML2PSSP transformation tool* uses mapping rules to map the meta-model of WSACML to the meta-model of the platform specific policies. Parallel to the product specific policy generation, the implementation of functional components like services, the portal frontend or the database layer is done by developers.

4.4 The Deployment Phase

Finally, the *platform specific security policies* are delivered to the *security administrator* for deployment. All artefacts which have no direct relation to the system's security like the implemented services or the contents of are portal are handled separately. With the help of the *policy import tool*, the generated *platform specific security policies* are imported into the security product. In general, new policies are imported directly whereas existing but changed policies are only updated. The security product will audit this changes within the policies for ensuring traceability.

5 Conclusion and Further Work

In this paper, we have described a model-driven development process for access control policies which is organised in four phases just like the traditional software development process (analysis, design, implementation, deployment). While we have described the development and application of the tools used in this process to concrete scenarios in previous work, we have focused on the overall structure of the access control policy development process and the tools' role as well as their interaction with the enterprise security infrastructure here. The mda-based approach results in a unified process for developing access control policies which helps bridging communication gaps between business and IT perspectives. At the same time, this streamlined development model may also make security development more efficient. A real business scenario for applying the proposed development process of access control policies is the opening of a current account in the banking domain. The business department models the business process and adds additional access control information. Refining them results in the secured business process model, from which all necessary information can be

extracted, so that WSACML policies and later product specific security policies for e.g. CA SiteMinder can be generated. A comprehensive version of this scenario is currently in progress and will be published in future work. Further work will be done on the integration of enterprise-wide compliance regulations within the development of access control policies as well as on methods to improve and measure the business process's security quality during its life cycle.

References

1. Neubauer, T., Klemen, M., Biffl, S.: Secure business process management: A roadmap. In: Proceedings of the 1st International Conference on Availability, Reliability and Security, IEEE Computer Society (April 2006) 457 – 464
2. Weske, M.: Business Process Management – Concepts, Languages, Architectures. Springer, Berlin (2007)
3. Object Management Group, Inc.: Model Driven Architecture (MDA) (July 2001)
4. Object Management Group, Inc.: MDA Guide – Version 1.0.1 (June 2003)
5. Frankel, D.S.: Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley, Indianapolis, Ind, USA (2003)
6. Emig, C., Brandt, F., Abeck, S., Biermann, J., Klarl, H.: An access control metamodel for web service-oriented architecture. In: Proceedings of the International Conference on Software Engineering Advances, IEEE Computer Society (August 2007)
7. Emig, C., Kreuzer, S., Abeck, S., Biermann, J., Klarl, H.: Model-driven development of access control policies for web services. In Khoshgoftaar, T., ed.: Proceedings of the 9th IASTED International Conference Software Engineering and Applications, Orlando, Florida, USA, IASTED (November 2008) 165–171
8. Klarl, H., Wolff, C., Emig, C.: Abbildung von Zugriffskontrollaussagen in Geschäftsprozessmodellen. In: Modellierung 2008 – Workshop Verhaltensmodellierung: Best Practices und neue Erkenntnisse, Berlin (March 2008)
9. Klarl, H., Wolff, C., Emig, C.: Identity management in business process modelling: A model-driven approach. In: 9. Internationale Tagung Wirtschaftsinformatik – Business Services: Konzepte, Technologien, Anwendungen, Band 1, Vienna, Austria, Österreichische Computer Gesellschaft (February 2009) 161–170
10. Lang, U., Schreiner, R.: Model driven security management: Making security management manageable in complex distributed systems. In: Modeling Security Workshop in Association with MODELS '08, Toulouse, France (2008)
11. Rees, J., Bandyopadhyay, S., Spafford, E.H.: PFIRES: a policy framework for information security. Communicaiton of the ACM **46**(7) (2003) 101–106
12. Rodríguez, A., Fernández-Medina, E., Piattini, M.: M-BPSec: A method for security requirement elicitation from a UML 2.0 business process specification. In: Advances in Conceptual Modeling – Foundations and Applications. Volume 4802 of Lecture Notes in Computer Science., Springer (November 2007) 106–115
13. Nagaratnam, N., Nadalin, A., Hondo, M., McIntosh, M., Austel, P.: Business-driven application security: From modeling to managing secure applications. IBM Systems Journal **44**(4) (2005) 847–867
14. Emig, C., Weisser, J., Abeck, S.: Development of SOA-based software systems - an evolutionary programming approach. In: Proceedings of the International Conference on Internet and Web Applications and Services, IEEE Computer Society (2006)