2009 International Conference on Services Computing

A Security Meta-Model for Service-oriented Architectures

Michael Menzel
Hasso-Plattner-Institute
Prof.-Dr.-Helmert Str. 2-3
14482 Potsdam, Germany
michael.menzel@hpi.uni-potsdam.de

Christoph Meinel
Hasso-Plattner-Institute
Prof.-Dr.-Helmert Str. 2-3
14482 Potsdam, Germany
meinel@hpi.uni-potsdam.de

Abstract

Service-oriented Architectures (SOA) facilitate the provision and orchestration of business services to enable a faster adoption to changing business demands. Several approaches have been described to generate executable description of service orchestrations based on visual business process models. These models describe workflows and related information on an abstract level supporting business analysts to state and verify business requirements.

In previous work, we have adopted this approach to simplify the security engineering in Service-oriented Architectures. We foster a model-driven approach based on the integration of security annotations in visual modelling notation. These annotations are gathered and translated to a domain-independent security model that facilitates the generation of enforceable security configurations (e.g. WS-SecurityPolicy).

In this paper, we introduce our security meta-model for SOA that constitutes the foundation for our model-driven approach. Based on a model for service interactions that describes the exchange of information in a service-based system, we define a model to express security requirements and policies, and introduce a mapping to WS-Policy and WS-SecurityPolicy.

1. Introduction

Service-oriented system engineering gains more and more attention, since IT-infrastructures evolved into distributed and loosely coupled enterprise system landscapes. Service-oriented Architectures (SOA) expose company's assets and resources as business services and enables the orchestration of these services in the scope of business processes. Therefore, business process modelling constitutes the foundation to describe, standardise and optimise organisational workflows and enables a faster adoption to market changes and business demands. A business process model can describe orchestrations of activities within organisations and complex interactions with business partners. In addition, related business requirements can be indicated on an abstract level.

In the domain of process-aware information systems, security and privacy intentions, legal regulations, and risk assessments define the requirements that specify how company assets such as systems, services and information have to be protected and how the availability of resources must be managed. Especially the cooperation with business partners – demanding the utilisation of services across organisational boundaries – constitutes the key aspect regarding security in Service-oriented Architectures, since the seamless and straightforward integration of cross-organisational services conflicts with the need to secure and control access. A broad range of access control models, security concepts and related implementations have emerged over the last decades and have been adapted in the scope of SOA to enforce security goals.

While some attributes of business processes, such as control- and data-flows, are directly expressed in the scope of the process models itself, related security goals are not expressed in this context. Currently available process modelling notations have not the ability to capture security intentions, authorisation concepts or to evaluate risks.

As we have described in previous work [25], business process modelling offers an appropriate layer to describe security requirements and to evaluate risks. In [14], we presented an enhancement for BPMN [7] to enable the assessment of risks based on the evaluation of assets and the trustworthiness of participants, and to enable the annotation of security requirements such as confidentiality or integrity. Similar approaches exist, for example as presented by Rodríguez [21], that provide extensions for BPMN to express basic security requirements. However, these approaches do not facilitate the creation of security policies that comply to the modelled security intentions. In [14], we described the process of a model-driven generation of security policies based on security patterns.

The model-driven generation of security configurations is an emerging research area. Similar approaches have been defined by Breu *et al.* proposing a methodology for security engineering in service-oriented Architectures [11], SecureUML [3] focusing on access control constraints, and UMLSec [13] as an extension for UML to model communication-based security goals. However, these approaches do not provide a holistic approach for Web Service security and do not describe a mapping to WS-SecurityPolicy [15] to enforce confidentiality, integrity, identification, and authentication.

In this paper, we present a meta-model for security in Service-oriented Architectures that is the central element in our model-driven approach: Information at the modelling layer is gathered and translated to this model that is used to facilitates the generation of specific security configurations (e.g. stated in WS-Policy). Our meta-model for security in Service-oriented Architectures

- describes the basic entities, relations and associated roles in a Service-oriented Architecture.
- provides an abstract policy model and defines specific security constraints for the security goals confidentiality, integrity, authentication, and identification.
- provides a mapping to WS-Policy and WS-SecurityPolicy.

The rest of this paper is organised as follows. In Section 2 we provide an overview about our model-driven approach to generate security policies based on modelled annotations in business processes. To support this approach, we introduce our basic model to describe entities such as service and their relations in an SOA in Section 3. The following section introduces security specific models to describe policies and security constraints. To prove the applicability of our model, Section 5 outlines the mapping to WS-SecurityPolicy. Section 6 describes related work, while the final section concludes this paper.

2. Modelling Security in Business Processes

Business process modelling provides an abstract view on workflows, information, and organisations. It supports business analysts and business process experts to analyse and improve processes concerning business requirements. To represent these processes, various visual modelling languages have been proposed in the recent years including the Business Process Modeling Notation (BPMN) [7] and the Unified Modeling Language (UML) [8]. Visual process models that are described using these visual notations can be translated to process descriptions that are executable by a process engine. In [18], an approach has been introduced to generate a Business Process Execution Language (BPEL) [2] description based on BPMN. BPEL is an orchestration language for Web Services and is based on the service model that is defined by the Web Service Description Language (WSDL) [4].

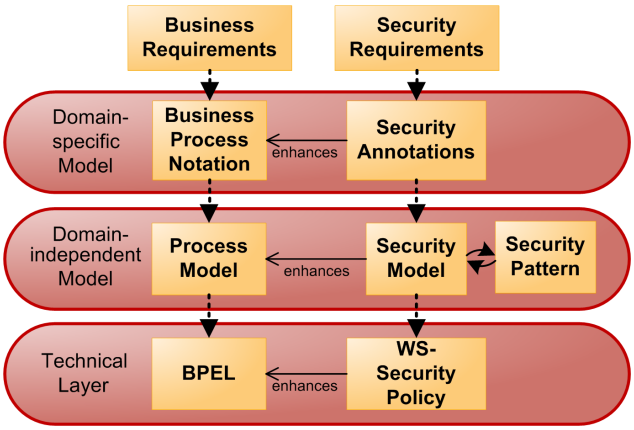

Figure 1. Model-driven generation of security policies

However, these approaches do not support the specification of security requirements in business process modelling and do not facilitate a generation of enforceable security configurations. Therefore, we introduced a model-driven approach to generate security policies in [14] that is illustrated in Figure 1. To state security requirements, we foster an annotation with security intentions in any domain specific visual modelling notation, for instance in BPMN as described in [14, 22, 26]. In the next step, information at the modelling layer are gathered and translated to a domain-independent security model. This security model relates to a formal process model (e.g. a Petri net) that can be used to verify the modelled security configurations.

However, the information gathered from the modelling layer does not have to be sufficient to generate enforceable security configurations, since further knowledge might be needed. Expertise knowledge might be required to determine an appropriate strategy to secure services and resource, since multiple solutions might exists to satisfy a security goal. For example, confidentiality can be implemented by securing a channel using SSL or by securing parts of transferred messages. To describe these strategies and their preconditions in a standardised way, we foster the usage of security patterns as described in [14]. Security patterns have been introduced by Yoder and Barcalow in 1997 [27] and are based on the idea of design patterns as described by Christopher Alexander [1]. Based on this work, we defined a pattern system that describes formalised patterns for each security goal and that is used to resolve appropriate security protocols and mechanisms. However,

a comprehensive security model as defined in this paper is required to express the preconditions and relations in a pattern system.

The final step in our model-driven approach is the transformation of security configuration into enforceable security policy languages, depending on the capabilities of the target environment.

3. A Model for Service Interactions

In this section, we will introduce the basic entities in our model and their relationships to describe an interaction in a service-oriented Architecture. Further, we will show how these entities can be mapped to Web Service specifications such as SOAP.

3.1. Security Base Model

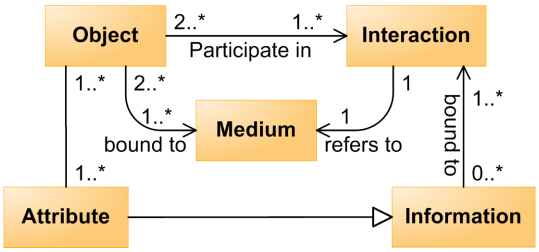

Figure 2. The Security Base Model

As introduced in [24], one of the basic entities in our model is an *object* that consists of a set of *attributes* and can participate in an *interaction*, see Figure 2. An interaction is always performed on a *medium* that is connected to the objects. For instance in the scope of Web Services, an object could be a Web Service client or a Web Service itself. A Web Service is bound to a medium – for example a TCP/IP network – and can interact with Web Service clients exchanging SOAP-Messages. Therefore, an interaction also involve the exchange of *information*.

To enable a detailed description of Web Service messaging, we model transferred information as *data transfer objects* as introduced by Fowler in [6]. A data transfer object is '*... little more than a bunch of fields and the getters and setters for them. [...] it allows you to move several pieces of information over a network in a single call. [...] the data transfer object is responsible for serializing itself into some format that will go over the wire.*' Figure 3 shows the adaptation of this concept to our model. A *data transfer object* represents serialised information and is an information itself. However, it can also contain information. This recursive structure facilitates the description of SOAP messages and its message parts. Figure 4 visualizes the mapping to

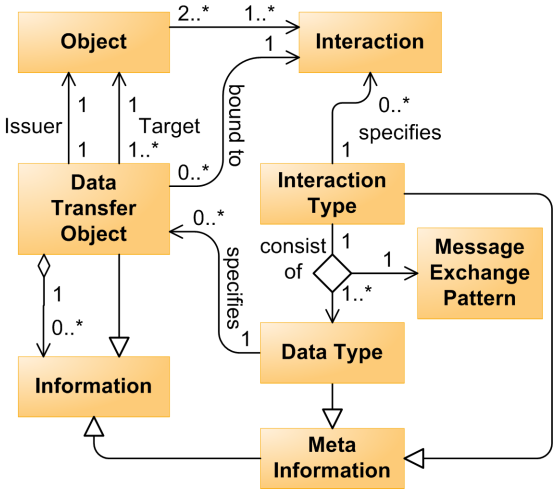

Figure 3. Modelling Message Exchange

the SOAP message structure as defined in the SOAP messaging framework specification [9]. A SOAP envelope is a data transfer object that can contain different message parts that are data transfer objects itself.

As shown in Figure 3, a data transfer object is specified by a *data type* that defines its structure (e.g. using XML Schema). Accordingly, the interaction is described by an *interaction type* that consist of an message exchange pattern (as described by WSDL [4]) and a set of data types.

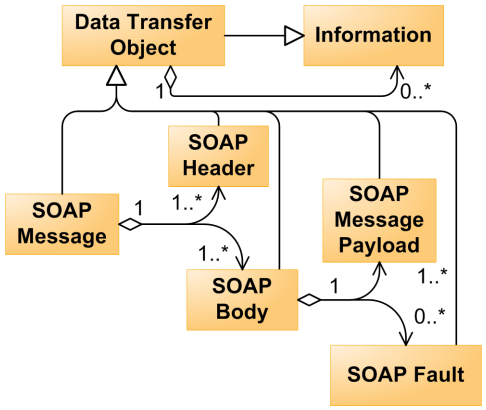

Figure 4. Mapping the model to SOAP

Moreover, a data transfer object has a *target* and an *issuer*. This reflects that a data transfer object can be send over several objects acting as intermediaries. Therefore, issuer and target do not have to correspond necessarily to the objects that are involved in an interaction exchanging a data transfer object. In the scope of Web Service technology, WS-Addressing [10] would be used to represent the issuer and target in a SOAP-message by including a SOAP header that is also a data transfer object.

3.2. Modelling Digital Identities

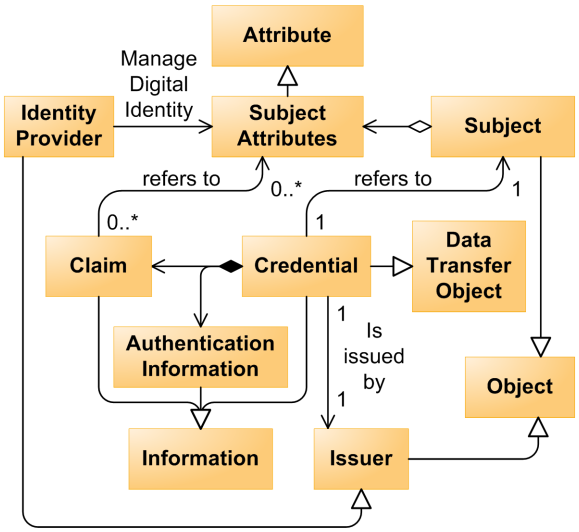

Figure 5. Modelling a Digital Identity

In our model, objects such as Web Service clients are entities in a technical system that operate on behalf of an organisation or person, which we refer to as *subject*. In the digital world, a subject is described by a set of *subject attributes* that are stored in an account managed by an *identity provider*, as shown in Figure 5. In reality, it is quite common that people (subjects) have multiple digital identities registered with different Identity Providers, for instance their employer, their email provider or various shopping sites in the internet.

In order to use services – whether Web Services or services in the internet – the exchange of identity information to identify, authenticate and authorise a subject is required. This information is represented by a *credential* that contains a set of *claims* [12] about the subject and an *authentication information*. A credential is created and asserted by an *issuer*, whose identity can be verified by the authentication information in the credential.

In a simple scenario, a user has to provide a username and a password to access a service. This credential (username/password) is issued by the user himself, contains a claim 'my user name is ...' and provides a password as authentication information. Furthermore, to enable a single sign-on in a more complex scenario, the credential might be a SAML token [19] issued by the user's employer (acting as identity provider). Such a token could contain arbitrary claims about the user (e.g. his role in the organisation) and includes a signature as authentication information to enable a verification of the token and its claims.

In summary, claims are made by an issuer about a subject and represent a set of subject attributes. While in a closed, administered, and trustworthy security domain the term security assertion is commonly used, the term claim has been introduced in the scope of the Web Service Specifications and is described by the identity meta system [12]. This term represents a degree of doubt regarding the brokering of identity information across trust domains in a loosely coupled system. The trustworthiness of a claim about a subject depends on the identity of the issuer and its reputation.

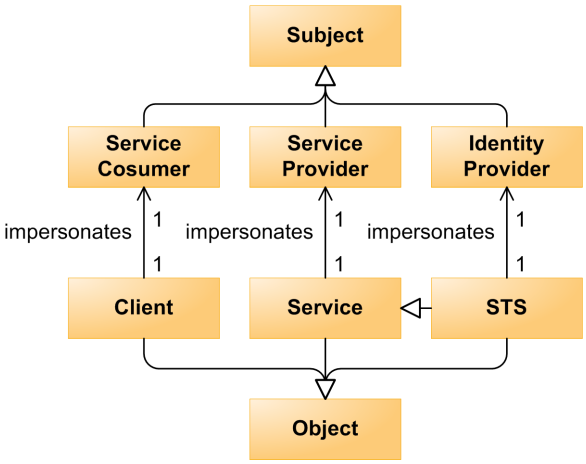

Figure 6. Roles in an SOA

As aforementioned, objects act in the digital world on behalf of subjects and, therefore, impersonate them. As shown in Figure 6, subjects and objects can act in different roles. *Services* are objects that offer capabilities on behalf of *service providers*, whereas a *service consumer* can interact with these services impersonating a *user*. An Identity provider is a subject that manages a digital identity of other subjects and is impersonated by a security token service (service interfaces are defined in [16, 19]), which is a specialisation of a service.

4. Modelling Security Requirements

In the previous section, we introduced the basic entities and their relations to model participants and interactions in a Service-oriented Architecture. Based on this general model, we will describe our approach to model security requirements in this section. Therefore, we will start with a basic model that will reveal the general structure of a security policy and its relation to other entities in our model. This generic description captures the essential policy elements to enable a mapping to any policy language. Based on this structure we will describe security constraints for specific security goals such as authentication and confidentiality.

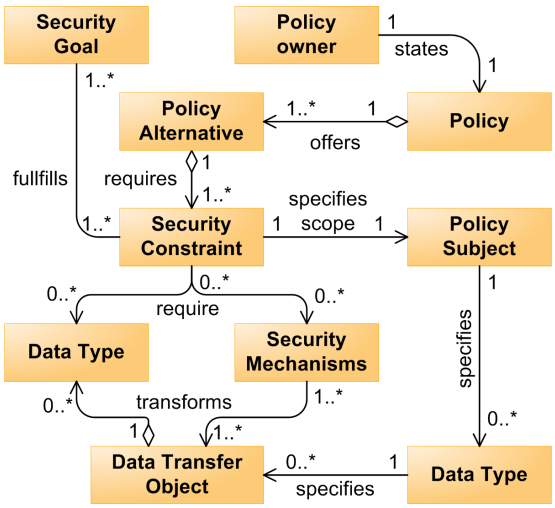

Figure 7. Security Policy Model

4.1. Security Policy Structure

As we have outlined in the previous section, the interaction between objects and the exchange of data transfer object are the important concepts to model communication in distributed and loosely coupled systems. Security policies in such systems define requirements to restrict the communication between participants in order to comply to predefined security intentions and organisational regulations.

A *policy*, as shown in Figure 7, is stated by a *policy owner* (e.g. a service) to express his requirements concerning the interaction with other objects. Therefore, we do not consider policies that relate to the internal functioning of an object. In our model, policy requirements always refer to interactions and related data transfer objects.

A policy consist of a set of *policy alternatives*. Each policy alternative requires a set of *security constraints* that describe requirements for a specific *security goal*. A policy will be fulfilled, if all security constraints of one policy alternative are enforced. Therefore, the usage of policy alternatives enables a disjunctive and conjunctive combination of security constraints and represents a disjunctive normal form. Since all logical formulas can be converted into the disjunctive normal form, any security policy structure based on a nesting of disjunctive and conjunctive combinations of policy requirements can be normalised and mapped to our model.

A security constraint can define two different types of requirements. First of all, a constraint can specifies the *security mechanisms* that must be applied to incoming or outgoing data transfer objects. A security mechanism specifies a protocol or an algorithm – e.g. for encryption and signature – that is used to transform data transfer objects. In addition, a security constraint can specify a set of *data types* that are required to be included in a data transfer object (e.g. a specific type of credential for authentication). Finally, a *policy subject* is associated with a policy constraint to specify the scope of the requirements concerning the data transfer objects. A security constraint applies to all data transfer objects that correspond to data types identified by the policy subject. In particular, a policy subject can refer to a specific data type (message policy subject) or all data types sent or received by a service or an operation (service/operation policy subject).

4.2. Specifying Security Goals

Our security policy model defines a general structure to group and describe security constraints in a distributed system. Based on this model, we have specified specialised constraints that define precisely the required information and security mechanisms regarding the security goals identification, authentication, confidentiality, and integrity.

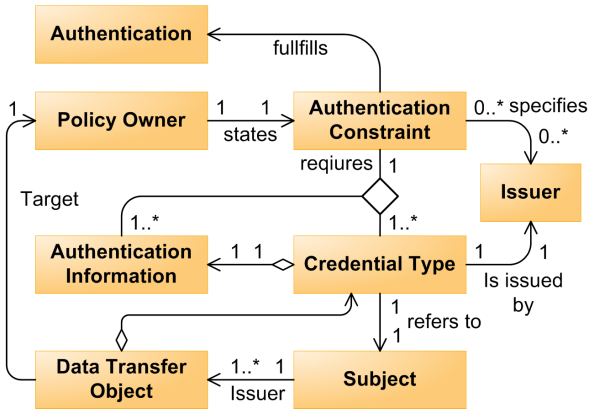

Figure 8. Authentication Constraint

Figure 8 illustrates the structure of an *authentication constraint*. This type of constraint requires a specific *credential type* (such as SAML or username/password) that must be included in corresponding data transfer objects. As aforementioned, the data transfer objects are identified by the policy subject as shown in Figure 7. In addition, an authentication constraint can specify an issuer, for instance a specific identity provider, that must have asserted this credential. The structure of the *identification constraint* is quite similar - instead of an credential type, this constraint specifies a required set of claim types to identify a user.

The confidentiality constraint is shown in Figure 9 and specifies

- a *security protocol* – This requirement identifies the protocol that is used to implement confidentiality (e.g. SSL to require a secure channel at the transport layer or WS-Security to secure the data transfer objects itself).

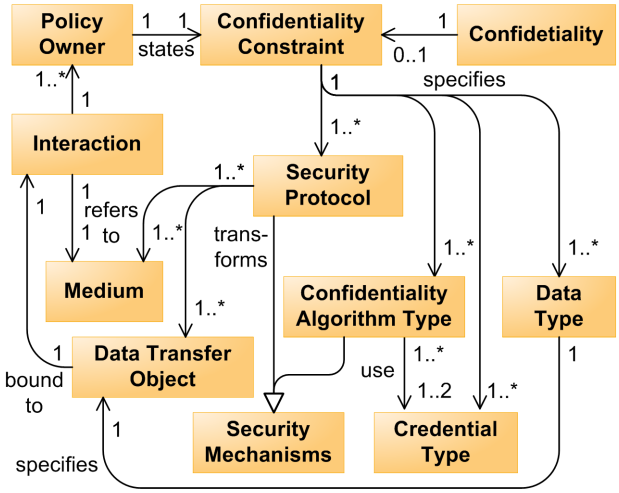

Figure 9. Confidentiality Constraint

- one or more *confidentiality algorithms* – One of the specified algorithm must be used by the protocol to encrypt information (e.g AES or DES).
- one or two *credential types* – identifies the type of credential that is used as a key to secure information. Depending on the specified algorithm, there must be a single credential type defined (symmetric encryption) or two types of credentials for incoming and outgoing data transfer objects (asymmetric encryption).
- one or more *data types* – defines the parts of the data transfer object that must be encrypted (e.g. a credit card number or a message header).

The structure and elements of the *integrity constraint* are similar to the confidentiality constraint and, therefore, is not shown in the course of this paper.

5. Mapping the Constraints to Security Policies

The security policy model and the security constraints introduced in the previous section describe the information that is needed to secure objects such as services. In this section we will introduce the mapping to WS-Policy and WS-SecurityPolicy that is used to generate security configurations in our model-driven approach.

5.1. Mapping to WS-Policy

WS-Policy [23] defines a grammar to group and express requirements and capabilities of Web Service entities that are referred to as policy assertions. However, WS-Policy does not specify the structure and semantic of policy assertions, since these are defined by additional specifications.

A WS-Policy expression consist of a set of policy alternatives representing a disjunction. Each alternative groups a set of policy assertions or additional alternatives and represents a conjunctive combination. WS-Policy provides an algorithm to convert a WS-Policy expression in a disjunctive normal form that enable a direct mapping to our model.

5.2. Mapping to WS-SecurityPolicy

WS-SecurityPolicy [15] defines a set of assertions for the usage in WS-Policy to express requirements concerning WS-Security, WS-Trust, and WS-SecureConversation. Since the confidentiality constraint is the most complex one in our model, we will describe the mapping to WS-SecurityPolicy using this example.

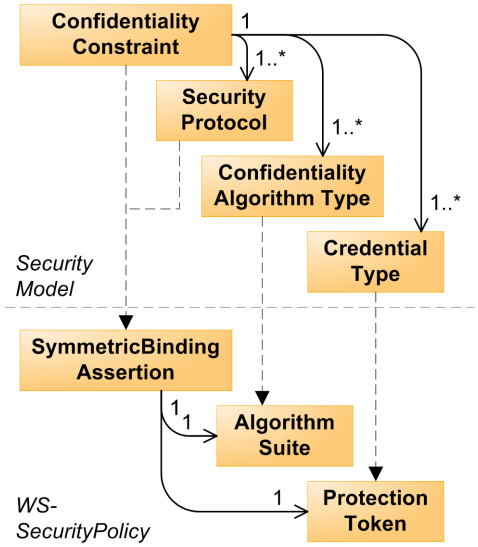

Figure 10. Mapping the Confidentiality Constraint to the SymmetricBinding Assertion

The core concept in WS-SecurityPolicy is the security binding that includes assertions to provide enough information to secure a message exchange. WS-SecurityPolicy defines three security bindings based on three security pattern:

- transport binding – requires a secure channel at the transport layer
- symmetric binding – encryption and signature with the same token for incoming and outgoing messages
- asymmetric binding – encryption and signature with different tokens for incoming and outgoing messages

The first step to generate WS-SecurityPolicy assertions, is the selection of a security binding that is determined by the protocol and the algorithm type in the confidentiality

constraint. For instance, the protocol SSL demands a transport binding, while WS-Security can be used with a symmetric or an asymmetric binding. In the latter case the selection depend on the algorithm type - for example, the usage of an symmetric encryption algorithm indicates the usage of a symmetric binding. Figure 10 shows the mapping to a symmetric binding assertion. The algorithm type can be mapped to a algorithm suite assertion, while the credential type can be translated to a protection token assertion.

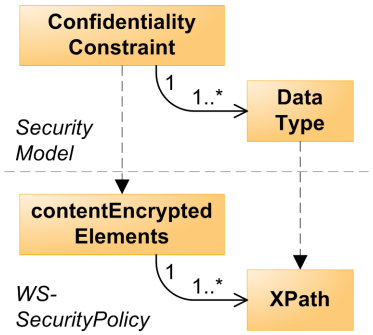

Figure 11. WS-SecurityPolicy Assertion to require encrypted message parts

WS-SecurityPolicy specifies an additional assertion called SymmetricBindingAssertion to define message parts that have to be protected based on a predefined binding assertion. Figure 11 shows the mapping for this assertion that is based on the entity data type in our model.

6. Related Work

Modelling security configurations in Service-oriented Architectures is an emerging topic. The OASIS consortium has published a public review draft describing a 'Reference Architecture for Service-oriented Architecture' [5] that provides a set of models and views to describe entities and roles in an SOA. Although the basic concepts are similar to our interaction model, it is clearly focused on high level concepts such as social structures and real world effects resulting from service invocations, while our model is designed to reveal dependencies to security related aspects on a technical layer.

Ruth Breu and Michael Haffner proposed a methodology for security engineering in service-oriented Architectures [11] that is based on a model-driven approach. Security requirements are modelled in a domain-specific language and transformed to a domain-independent language that is used to generate security policies. Their approach provides a generic framework that can be used to express security goals in domain-specific languages. In addition, they outlined a transformation to authorisation constraints. Although providing a generic framework, they do not considers specific Web Service characteristics such as claim-based identities and do not describe a mapping to WS-SecurityPolicy.

Model-driven security automating the generation of security configurations has been a topic of interest in recent years. For instance SecureUML [3] is a model-driven security approach for process-oriented systems focusing on access control. Similar to SecureUML, Jürjens presented the UMLSec extension for UML [13] in order to express security relevant information within a system specification diagram. One focus of UMLSec lies on the modelling of communication-based security goals, such as confidentiality, for software artefacts, while SecureUML describes desired state transitions and access control configurations for server-based applications. Both do not describe the link to business processes and related security requirements in a Service-oriented Architecture.

In the context of business processes, previous work done by Nagaratnam *et al.* [17] discusses an approach to express security requirements and how to monitor and manage them on the different enterprise architecture levels. This intention does not provide a detailed analysis of security goals, their conceptual models, and their relationship to the business process related entities. This issue has been addressed by Rodrguez *et al.* [20, 22] by defining a meta-model that links security requirement stereotypes to activity elements of a business process and proposed graphical annotation elements to visually enrich the process model with related security requirements. Although they support several security intentions, they do not provide a comprehensive security model to support a model-driven generation of security policies.

7. Conclusions

The model-driven generation of security configurations promises a simplified creation and management of security policies that comply to abstract security intentions stated and verified at the business process layer. Security requirements and parameters are translated to a security model that can be mapped to different policy specifications to generate enforceable security configurations.

In this paper, we introduced a security meta-model for SOA that constitutes the foundation for our model-driven approach. This model describes the basic entities, relations and associated roles (such as service and service consumer) in a service-oriented Architecture and provides the foundation to model interactions and the exchange of information. To describe the brokering of identity information, our model includes participants and claim-based digital identities.

We introduced a policy model to group and attach security requirements and outlined the effect to objects, data transfer objects and interactions. To express specific secu-

rity requirements, we described security constraints for the security goals confidentiality, integrity, authentication, and identification. Moreover, we described a mapping to WS-Policy and WS-SecurityPolicy to prove the applicability of our model.

7.1. Future Work

We stated that our proposed security model is a suitable foundation to describe and implement a model-driven transformation of abstract security intentions to enforceable security configurations in different application domains. As described in [14], security patterns are a promising approach to resolve additional information needed in the transformation process. In the next step we will use the proposed model for specifying the preconditions of the security patterns to enable an automated reasoning.

While WS-SecurityPolicy is used to express requirements of the service, requirements concerning the exchange of information can not be expressed in a standardised way. Therefore, we will investigate the integration of security constraints in BPEL.

References

[1] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobsen, I. Fiksdahl-King, and S. Angel. *A Pattern Lanuage: Towns - Buildings - Construction*. Oxford University Press, 1977.

[2] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guzar, N. Kartha, C. K. Liu, R. Khalaf, D. Knig, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. Web service business process execution language version 2.0. Specification, April 2007.

[3] D. Basin, J. Doser, and T. Lodderstedt. Model Driven Security for Process-Oriented Systems. In *SACMAT '03: Proceedings of the 8th ACM symposium on Access control models and technologies*, 2003.

[4] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana. Web services description language (wsdl) version 2.0. Specification, June 2007.

[5] J. A. Estefan, K. Laskey, F. G. McCabe, and D. Thornton. Reference architecture for service oriented architecture version 1.0. Public Review Draft, April 2008.

[6] M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[7] O. M. Group. Business process modeling notation, v1.1. Specification, January 2008.

[8] O. M. Group. Omg unified modeling language version 2.2. Specification, Februar 2009.

[9] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon. Soap version 1.2 part 1: Messaging framework (second edition). Specification, April 2007.

[10] M. Gudgin, M. Hadley, and T. Rogers. Web services addressing 1.0. Specification, May 2006.

[11] M. Hafner and R. Breu. *Security Engineering for Service-oriented Architectures*. Springer, October 2008.

[12] M. B. Jones. The identity metasystem: A user-centric, inclusive web authentication solution. Technical report, Microsoft Corporation, Microsoft Corporation One Microsoft Way, Building 42/4812 Redmond, WA 98052 USA, March 2006.

[13] J. Juerjens. UMLsec: Extending UML for Secure Systems Development. In *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 412–425, 2002.

[14] M. Menzel, I. Thomas, and C. Meinel. Security requirements specification in service-oriented business process management. In *ARES*, 2009.

[15] A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist. Web services security policy language 1.2. Public Draft Specification, Juli 2007.

[16] A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist. Ws-trust 1.3. OASIS Standard, March 2007.

[17] N. Nagaratnam, A. Nadalin, M. Hondo, M. McIntosh, and P. Austel. Business-driven application security: From Modeling to Managing Secure Applications. *IBM Systems Journal, Vol 44, No 4*, 2005.

[18] C. Ouyang, W. M. van der Aalst, D. Marlon, ter Hofstede, and A. H.M. Translating BPMN to BPEL. In *BPM Center Report BPM-06-02*, 2006.

[19] N. Ragouzis, J. Hughes, R. Philpott, and E. Maler. Security assertion markup language (saml) v2.0 technical overview, 2006.

[20] A. Rodríguez, E. Fernández-Medina, and M. Piattini. Towards a uml 2.0 extension for the modeling of security requirements in business processes. In *TrustBus*, pages 51–61, 2006.

[21] A. Rodríguez, E. Fernández-Medina, and M. Piattini. A bpmn extension for the modeling of security requirements in business processes. *IEICE Transactions*, 90-D(4):745–752, 2007.

[22] A. Rodriguez, E. Fernandez-Medina, and M. Piattini. Towards cim to pim transformation: From secure business processes defined in bpmn to use-cases. In *BPM*, pages 408–415, 2007.

[23] A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, and mit Yalinalp. Web services policy 1.5 - framework. Specification, September 2007.

[24] C. Wolter, M. Menzel, and C. Meinel. Modelling security goals in business processes. In *Proc. GI Modellierung 2008*, number ISBN 978-3-88579-221-5. GI LNI, Berlin, Germany, 1008.

[25] C. Wolter, M. Menzel, A. Schaad, P. Miseldine, and C. Meinel. Model-driven business process security requirement specification. *Journal of Systems Architecture Special Issue on Secure Web Services*, 2008.

[26] C. Wolter and A. Schaad. Modeling of task-based authorization constraints in bpmn. In *BPM*, pages 64–79, 2007.

[27] J. Yoder and J. Barcalow. Architectural patterns for enabling application security. In *PLoP*, 1997.