Domain-Specific Multi-modeling of Security Concerns in Service-Oriented Architectures

Juan Pedro Silva Gallino, Miguel de Miguel, Javier F. Briones, and Alejandro Alonso

Universidad Politécnica de Madrid (UPM)
{psilva,mmiguel,jfbriones,aalonso}@dit.upm.es

Abstract. As a common reference for many in-development standards and execution frameworks, special attention is being paid to Service-Oriented Architectures. SOAs modeling, however, is an area in which a consensus has not being achieved. Currently, standardization organizations are defining proposals to offer a solution to this problem. Nevertheless, until very recently, non-functional aspects of services have not been considered for standardization processes. In particular, there exists a lack of a design solution that permits an independent development of the functional and non-functional concerns of SOAs, allowing that each concern be addressed in a convenient manner in early stages of the development, in a way that could guarantee the quality of this type of systems. This paper, leveraging on previous work, presents an approach to integrate security-related non-functional aspects (such as confidentiality, integrity, and access control) in the development of services.

1 Introduction

Service-Oriented Architectures (SOA), of great popularity nowadays, and Web Services (WS), the technology generally used to implement them, go hand in hand in such a way that they are even referred to as WSOA (Web Service Oriented Architecture). They achieve the integration of heterogeneous technologies, providing interoperability, and yielding the reutilization of pre-existent systems.

At the same time, Model Driven Applications (MDA) [14] arises as a new paradigm that tries to shift models out of a mere documentary role, and turn them into a first class development artifact. This approach provides, among other benefits, a greater understanding of the system as a whole and a platform-independent development, improving the reusability of the design, and simplifying the evolution of the system, thus increasing productivity.

WS-* standards provide a strong foundation for the development of services. WS-Policy and related specifications offer the possibility of considering extra-functional concerns of those services. However, these specifications are XML-based, and its syntax is not designed to be read/written by humans. An abstraction of such languages is desirable.

From the experience obtained during the participation in a research project focused on the service-oriented reformulation of a banking core [3], the need

for means to independently address non-functional properties (NFPs) to foster design (functional and non-functional) reuse along different companies became evident. Different modeling techniques and approaches in use among the different financial institutions made input-models-language independence fundamental. Moreover, the secrecy of security designs is also paramount; security design should only be accessed by trusted number of individuals. Different banks use different infrastructures, and different security solutions and approaches, for instance. It is the authors' belief that this situation repeats itself in other industries.

If the research's results were to be useful throughout the banking industry, the NFPs of the systems had to be supported in different shapes and forms. A clear candidate approach to address this issue, therefore, is that proposed by Multi-Dimensional Separation of Concerns (MDSoC) [21], and seconded in Viewpoints [7], aspect-oriented modeling (AOM) or early aspects (EA) [5,17]. These approaches share common ideas, and allow for the separation of functional and extra-functional characteristics of the systems *at design time*, providing a base for the solution presented in this paper.

Within these different views or concerns of the system, the elements have to be described somehow. In general, aspect-oriented solutions prefer general purpose modeling languages (GPMLs) for this task. GPMLs' approach towards this problem is to provide extension mechanisms to adapt the generic modeling elements to the particular concern or domain that is being addressed by the model. Domain-specific modeling languages (DSMLs), however, takes a different approach, and reduce the available modeling elements to key, familiar problem-domain concepts (and rules), narrowing down the design space [8], to achieve higher levels of productivity. In the approach presented in this paper, DSMLs have been selected to appropriately address the different NFPs.

A final element of concern is the technical expertize needed to express correct (both syntactically and conceptually) security and other NFPs' configurations. One of the ideas in SOA is that business be aligned with the IT infrastructure. Therefore, stakeholders at the business level are candidate to providing input models of the system. Nevertheless, these stakeholders, although capable of expressing security intentions or requirements to these business services, are probably not security savvy enough to express a complete security configuration, nor are aware of the availability of the different security infrastructure elements.

Although multiple implementation technologies exist to facilitate the development of web services and SOA systems, the lack of a sound methodological base for the development of such applications stresses the need for new modeling methods or techniques that could guarantee the quality of the development of this type of systems. In summary, it was identified that there exists a lack of a design solution that:

- Allows an independent development of the functional and non-functional (NF) concerns of service-oriented architectures, fostering reuse.
- Permits that each concern be addressed in a convenient manner, at the appropriate abstraction level.

- Avoids the need for non-functional (in this case, security) expertize at the business/system architect role.
- Is modular enough so that the functional model is not polluted with non-functional information, and secrecy of security design could be maintained.
- Provides a practical way of measuring and analyzing the system for non-functional characteristics in early stages of the development.

Some approaches applying MDA to the development of Web Services already exist (e.g. [16], [10]). However, these approaches do not offer thorough support for access control (AC) descriptors, code generation, Web Services Description Language (WSDL), and WS-(Security)Policy. This paper, leveraging on previous work, introduces the CIM-to-PIM layer of an approach to integrate security-related non-functional aspects (such as confidentiality, integrity, and AC) in the development of services, parting from high-level descriptions of business services compositions, and incorporating technical elements through a chain of transformation and compositions, towards the final implementation of the individual, security-aware services.

The paper is structured as follows. Section 2 describes the general architecture of the approach, while section 3 introduces the Supply Chain Management [25] use case, and describes its implementation along the steps of the proposed solution. Section 4 presents some relevant related work. Finally, section 5 provides some conclusions and possible future lines of work.

2 Integrating the Non-functional Concerns

As first presented in [19], a complete solution addressing non-functional aspects of service architectures was defined. The addressed characteristics are NF properties subject of being described as policies, of which security and access control are prominent examples. This section presents the methodology proposed to integrate these non-functional aspects in the development of web services. A well-known use case, including security concerns, has been selected to illustrate the usefulness of the approach.

2.1 A Framework for the Inclusion of Security Concerns

Figure 1 shows the overall structure of the designed solution, consisting of a chain of model transformations and compositions (indicated by arrows). Every box in the figure indicates a different type of model, including:

- Business model indicating non-functional intentions (CIM Level).
- Functional system model (PIM 1 Level).
- Non-functional properties models (PIM 1 Level).
- Intermediate Meta Model (iMM) model (PIM 2 Level).
- WSDL, WS-Policy, and XML models (PSM Level).

The iMM metamodel was defined with the objective of achieving an abstraction both of input modeling techniques/metamodels, and output target platforms. It is also designed for holding, in one unique model, all information necessary for analysis and to generate the different output artifacts. The different steps' resulting artifacts, including the iMM models, are automatically generated, and the developers need never interact with them if unwanted. However, the modeler has the opportunity to tune/modify the intermediate results to shape the subsequent outcome.

The presented approach proposes the development of each individual concern as an independent model, in a similar fashion to that of Multi-Dimensional Separation of Concerns [21]. Different domain-specific metamodels, each one appropriate to the particular concern in hand, are used to define the non-functional models. The addressed security concerns are individually mapped from the iMM into implementation artifacts in the target platform, independently of the particular services under design.

In that way, the system under implementation provides the particular values for the final implementation elements, but concern models (and their mappings onto the platform) may evolve independently. Such models are composed into a complete model of the system by weaving them together. A possible set of metamodels for the consideration of security concerns in service-oriented architectures is presented in [19] and included in section 2.3.

For the generation, the process model is split into atomic services and their associated NF intents. The different resulting outputs to be obtained, indicated in figure 1, include access-control descriptors (platform dependant), code templates (platform dependant), and WSDL and WS-Policy documents (common to all target platforms). The composition execution should then make use of these artifacts to execute the process.

2.2 Development Process

The sequence of steps to perform in the proposed approach, leveraging on that first introduced in [19], is presented here:

1. A business process model, annotated with abstract security intents (also known as primitives), is presented as input to the process.
2. The intents at CIM level drive the selection of the appropriate security and access-control models. The developer selects, tunes, or even defines new AC and security models, as better fits the system.
3. A functional PIM model is derived from the CIM model or, alternatively, provided by the modeler as an input functional model at PIM level (PIM 1).
4. By means of composition models, the modeler indicates which non-functional properties should be associated to which resources in the functional model. Ideally, pre-filled composition models should be proposed by a tool automating the development process.
5. At this stage, the platform-independent iMM model (PIM 2) contains all (functional and extra-functional) information.

6. Next, the iMM model is transformed into platform dependant (PSM) models. Default values may be used for the generation, unless optional platform information (indicating preferences and/or availability of platform characteristics) is fed to the transformation. In the example presented in this paper, assertions in a WS-SecurityPolicy metamodel, extended with a "preference" attribute, may be used for representing the available/preferred mechanisms to use from the target platform.
7. Finally, service and policy descriptors are derived from the PIM 2 model. Resulting WSDL documents hold references to the appropriate policies defined in the different generated WS-Policy documents. Simultaneously, code templates and non-functional configurations are automatically created.

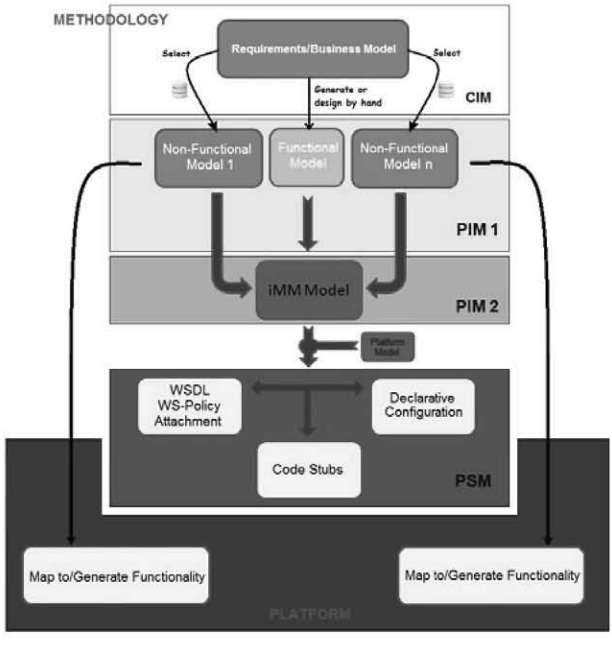

Fig. 1. Structure of the proposed solution

Within the process model, the modeler may express high level security intents (e.g., non-repudiation, confidentiality, integrity), instead of technical mechanisms which are necessary to achieve them (e.g., encryption, signature, time-stamping). The know-how of the mechanisms necessary to achieve the intents is contained in the metadata and the transformations. In this way, the business/system architect is freed from the need of technical security knowledge.

Repositories containing different expert-made AC and security models could provide the non-functional inputs. The idea behind the approach is that different NF models, suitable for different concerns in different domains, would represent

a pool of options from which to select the most appropriate one. Experts in the particular NF domain create such models, and also indicate how the different elements map onto the target platform. Adequate metadata is associated to these models for automatic discovery support. In that way, appropriate models are automatically selected from the repositories and offered to the modeler.

Analyses and metrics, performed and calculated on the iMM model, reassure the modeler confidence in the correctness of the design. Errors detected in this stage are less costly to correct, and the system architect may have the opportunity to consult a domain expert to solve any complicated issue.

The benefits of this approach is twofold: the system's developer need not be security savvy, and allows greater reuse of (functional and non-functional) designs. NF models may be reused among different projects. Also, different NF characteristics may be applied to the same functional design to generate different resulting systems.

2.3 Metamodels

Referring to the metamodels employed in this solution, some pre-existent metamodels have been adopted, some other were merged together, whilst the rest had to be defined for the occasion. The different metamodels presented here, are described in more detail in [19].

Modeling Security Intents at the CIM Level. Business processes diagrams have become the mainstream alternative to model the compositions and interactions of services. Such models provide a great means for describing the interaction of the services at a high level of abstraction. They also represent an outstanding instrument on which to express abstract non-functional intentions that allow stakeholders to express their security concerns without the need for a detailed technical knowledge. Ontologies have risen as useful tools to agree on a common language for the description and discovery of elements (web services, for instance). The use of an ontology at the CIM level may provide not only a means to define a common vocabulary for security intents, but may also represent a tool for the matchmaking [1] of the non-functional concerns models contained in the repositories that may fulfill the expressed requirements.

In the security domain, the NRL ontology [9] represents a very complete option, scoping from high-level security objectives, through security credentials, to technical encryption algorithms, and has been selected to formalize a vocabulary for the security intents.

Functional Input Metamodel: UML. If a PIM level input is to be provided, UML, the modeling standard of choice in most cases nowadays, has been selected as the input's metamodel. UML's SoaML profile [2] was implemented and applied to these input models to guide the UML-to-iMM transformation.

[1] The process of identifying the elements that fulfill the imposed requirements, and its degree of fulfillment.

Security Metamodel: QoS Metamodel. UML's QoS profile [22] defines the "QoS Framework Metamodel" as an extension to UML's metamodel. The defined concepts can be used to model non-functional characteristics as a domain-specific metamodel (as in, for instance, [10]). This is the metamodel selected to illustrate the use of the approach being presented.

Access Control Metamodel: SecureUML. SecureUML [11], a Role Based Access Control (RBAC, one of the currently most used AC techniques) metamodel is used to exemplify possible access-control input models. The modeling and implementation of AC, which follows a similar process to that of the security properties, is not included in this article.

Weaving Metamodels. Weaving metamodels guide the composition mechanisms of the different models. In this case, the composition of functional models with AC models or security models require different weaving associations, described in [19]. The semantics of the associations are implemented as part of the composition rules.

iMM Metamodel. The intermediate metamodel is one of the fundamental parts of this approach. Composed of three differentiated parts (System Design, Access Control, and Policy), this particular implementation, described in [19], was designed to be as general as possible, in order to maximize support for AC techniques and policy standards.

The functional part of the metamodel follows a component-based approach, and is used to describe service components, entities and interfaces, among other metaclasses. The generic metamodel from [15], incorporated in the "Access Control" part, allows for the use of different access-control techniques (and mutation among them). Finally, the "Policy" package of CBDI-SAE Meta Model for SOA [4] was selected for the service policies part. The three selected metamodels were then studied to identify equivalent concepts that provided merging points.

An integrated metamodel, such as iMM, provides a great subject of analysis and metrics' calculations. In this particular case, access control conflicts analysis, or security coverage metrics, for instance, could be defined. Analyses and metrics for iMM models are out of the scope of this paper.

WSDL Metamodel. A WSDL metamodel has been used for the transformations, and in the compositions (in order to visualize the future WSDL document as a model and facilitate the specification of policy application points).

WS-Policy Metamodel. Equivalently, a WS-Policy metamodel has been developed. The transformations and later generation of WS-Policy documents operate on this metamodel, including the case of WS-SecurityPolicy models.

WS-SecurityPolicy Metamodel. A WS-SecurityPolicy metamodel has been defined as an extension to the WS-Policy metamodel, providing a means to edit and validate security policy assertions, and to indicate the platform's security

capabilities and preferences. It is, to the best knowledge of the authors, the first available implementation of a WS-SecurityPolicy metamodel.

3 WS-I's Supply Chain Management Use Case

To illustrate the use of the proposed approach, the Web Services Interoperability Organization's (WS-I) [23] Supply Chain Management (SCM) use case [25] has been selected.

The SCM application [25,24] presents a typical business-to-consumer (B2C) model: a *Retailer* offering electronic goods to consumers. To fulfill orders the *Retailer* has to request products ("shipGoods" operation) and manage stock levels in the three *Warehouses*. When an item in stock falls below a certain threshold, the *Warehouse* must restock the item from the relevant *Manufacturer*'s inventory ("POSubmit" operation, a typical business-to-business (B2B) model). A complete description of the use case architecture's can be found in [25], and in [26] for security requirements. Access-control was originally not addressed in this use case, and will not be included in the example in this paper.

Two frameworks for the Java programming language have been selected to support the execution of the system:

- Spring framework [20], a framework leveraging enterprise-level functionalities for Java POJOs (Plain Old Java Objects).
- Apache CXF [1] framework, supporting policies and Literal and RPC style Web Services (as demanded by the SCM use case specification).

Frameworks provide much functionality which would be otherwise necessary to generate for the execution of the different systems, greatly simplifying the development of generators. Newer frameworks' declarative-based configuration, in particular, support the configuration of non-functional characteristics of the services (such as security or AC, in this case) independently of the functional code. This is of great usefulness in this case, allowing for an independent generation of the different artifacts, and the reuse of generators for different target platforms.

3.1 Creating and Selecting the Input Models

A process model of the system is considered as the primary input. This model, annotated with security and access control intents, indicates the non-functional requirements on the different services. For instance, an interaction marked "non-repudiation" would imply associating to the message the actions of authenticating, signing and time-stamping. The NRL ontology [9], designed to facilitate automatic discovery and invocation, has been selected in this example to formalize the vocabulary for the annotation. Using NRL's matchmaking algorithm, different matching security and AC models are presented to the modeler for selection. In this case, the QoS model would match the different security required characteristics, as it models the security mechanisms to satisfy them.

Table 1. QoS Catalog for the QoS Category "Security"

	Dimension	Unit Type	shipGoodsRequest
Int	IntegrityAlg.	Enum. Literal	SHA1
	TimestampRequired	Boolean	true
	EncryptSignature	Boolean	true
	Signature Encryption Algorithm	Crypto Alg.	Crypto Alg.'s value
Conf	Crypto Alg.	String	AES256
Auth	TokenType	Enum. Literal	X509
	TokenKey Alg.	Enum. Literal	RSA15
	SignToken	Boolean	true
	EncryptToken	Boolean	true

Retailer System's UML Model. Functional models of the system will most likely be derived from the process model, developed from scratch to satisfy it, or reused from a previous system model (perhaps even reverse-engineered from legacy code). The 'de facto' standard for functional systems' modeling is UML, so it was decided to use it to model the Retailer system. As described in section 2.3, the SoaML profile is used to guide the posterior transformation. UML class diagrams for all the services in the SCM example are available in [25].

Modeling of the Security Intents with the QoS Metamodel. The first step towards modeling security concepts under the QoS metamodel is defining a QoS Catalog for the target (one or more) QoS Category. A catalog defines the different characteristics (may be regarded as meta-classes in a metamodel) and dimensions (the variables defining a particular characteristic) that may be present in a model of such category. In this particular case, only one category will be modeled: Security. Within this category, three characteristics are defined: Integrity, Confidentiality, and Authentication. Table 1 presents the different QoS Dimensions defined for these three characteristics, and the values assigned to them in the case of the "shipGoodsRequest" message of the "shipGoods" operation, offered by the *Warehouse* and used by the *Retailer* services.

3.2 Transformation into the iMM Model

Composition of Retailer System's UML Model with Non-Functional Models. Having two different non-functional aspects of the system under consideration, two composition models are necessary. The first one relates the *Retailer* system elements with the security characteristics in the QoS model.

Figure 2 presents the three panel view provided by the AMW tool [6] to create such composition models. In this figure, the different model elements are associated to its security requirements. The figure brings out, for instance, the association of "ShipGoodsResponse" with the security characteristics of signing, with the corresponding *Warehouse* X.509 certificate, the message and timestamp. The composition mechanism for SecureUML input models is equivalent.

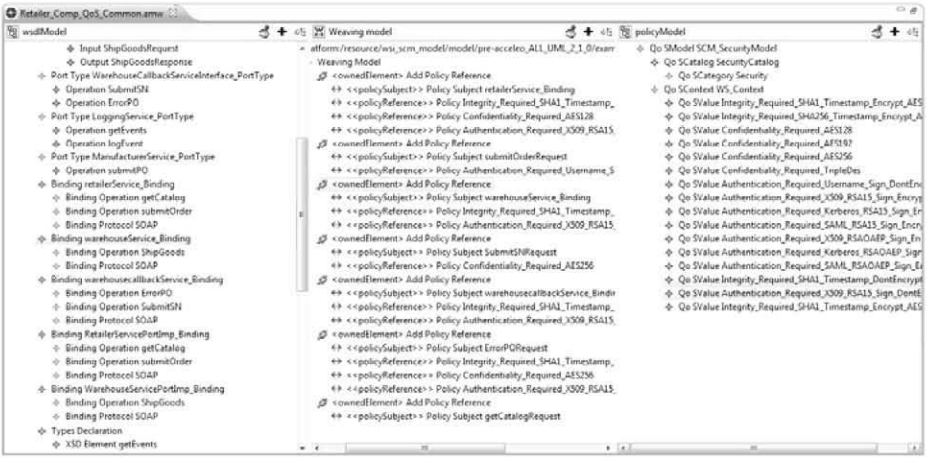

Fig. 2. *Composition of Retailer's QoS and UML Models*

3.3 Generation of Resulting Artifacts

For the generation of both WSDL and WS-Policy documents, the models are first transformed into a XML model and later extracted into the final XML syntax. The resulting WSDL documents (not included, equivalent to those in [25]) reference, within the description of the service elements, the corresponding service policies. The resulting WS-Policy document (figure 3) contains the policies applied to the *Warehouse* service (correspondent to the requirements and the QoS input model) and provide an equivalent expression, in WS-SecurityPolicy assertions, to the security configurations described below. The description of the generation of the functional code is out of the scope of the paper.

Generation of the WS-Security Configuration. The following configuration elements represent the target platform's results for the non-functional concerns requirements. All information necessary to generate the non-functional configuration files is available in the iMM model.

Figure 4 shows an excerpt of the configuration file generated for the Warehouse service. In it, the "WarehouseA" service is defined as a *jaxws:endpoint* and configured correspondingly. The actual bean implementing the service ("warehouseAService") is passed as a reference to the endpoint. The implementor bean is itself configured with the different properties it requires.

Security interceptors are also defined for the service endpoint. Only the output interceptor ("WarehouseEndTimestampSign_Response") is presented in figure 4 for visualization. The input interceptor's configuration is very similar, and therefore excluded. Security interceptors' configuration represent, therefore, the actual piece of the configuration file that depends of the security policy part of the iMM model.

```
<?xml version = '1.0' encoding = 'UTF-8' ?>
<wsp:Policy xmlns:sp = 'http://schemas.xmlsoap.org/ws/2002/12/secext'
            ...
            xmlns:wsp = 'http://www.w3.org/ns/ws-policy'>
  <wsp:Policy name = 'warehouseService_Binding_Binding_Policy' wsu:Id = 'warehouseService_Binding_Binding_Policy'>
    <wsp:All>
      <sp:AsymmetricBinding wsp:Ignorable = 'false' wsp:Optional = 'false'>
        <sp:IncludeTimestamp wsp:Ignorable = 'false' wsp:Optional = 'false'/>
        <sp:InitiatorToken wsp:Ignorable = 'false' wsp:Optional = 'false'>
          <wsp:Policy name = 'InitiatorTokenNestedPolicy' wsu:Id = 'InitiatorTokenNestedPolicy'>
            <wsp:All>
              <sp:X509Token wsp:Ignorable = 'false' wsp:Optional = 'false'
                  sp:IncludeToken = 'http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient'>
                <wsp:Policy name = 'InitiatorTokenX509TokenNestedPolicy' wsu:Id = 'InitiatorTokenX509TokenNestedPolicy'>
                  <wsp:All>
                    <sp:WssX509V3Token10 wsp:Ignorable = 'false' wsp:Optional = 'false'/>
                  </wsp:All>
                </wsp:Policy>
              </sp:X509Token>
            </wsp:All>
          </wsp:Policy>
        </sp:InitiatorToken>
        <sp:RecipientToken wsp:Ignorable = 'false' wsp:Optional = 'false'>
          <wsp:Policy name = 'RecipientTokenNestedPolicy' wsu:Id = 'RecipientTokenNestedPolicy'>
            <wsp:All>
              <sp:X509Token wsp:Ignorable = 'false' wsp:Optional = 'false'
                  sp:IncludeToken = 'http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToInitiator'>
                <wsp:Policy name = 'RecipientTokenX509TokenNestedPolicy' wsu:Id = 'RecipientTokenX509TokenNestedPolicy'>
                  <wsp:All>
                    <sp:WssX509V3Token10 wsp:Ignorable = 'false' wsp:Optional = 'false'/>
                  </wsp:All>
                </wsp:Policy>
              </sp:X509Token>
            </wsp:All>
          </wsp:Policy>
        </sp:RecipientToken>
        <sp:AlgorithmSuite wsp:Ignorable = 'false' wsp:Optional = 'false'>
          <wsp:Policy name = 'AlgorithmSuiteNestedPolicy' wsu:Id = 'AlgorithmSuiteNestedPolicy'>
            <wsp:All>
              <sp:Basic128Rsa15 wsp:Ignorable = 'false' wsp:Optional = 'false'/>
            </wsp:All>
          </wsp:Policy>
        </sp:AlgorithmSuite>
      </sp:AsymmetricBinding>
      <sp:SignedParts wsp:Ignorable = 'false' wsp:Optional = 'false'>
        <sp:Header wsp:Ignorable = 'false' wsp:Optional = 'false' Name = 'ConfigurationHeader'
                Namespace = 'http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Warehouse'/>
        <sp:Body wsp:Ignorable = 'false' wsp:Optional = 'false'/>
      </sp:SignedParts>
    </wsp:All>
</wsp:Policy>
```

Fig. 3. *Warehouse Service's WS-Policy document*

Within this configuration, and encircled, two entries stand out:

- the different securing actions to be performed (key="action" and, according to the requirements, value="Timestamp Signature"),
- the actual elements to be signed (key="signatureParts" and, again following the requirements, value="..Timestamp; ..Body; ..Header").

The presented use case, although briefly described, illustrates how the proposed approach provides mechanisms to address the objectives placed in section 1. Abstract NF intents are first mapped to NF design models, composed into a complete system model, refined, and finally mapped to implementation artifacts. The security interceptors and the access control context's elements (not presented) represent the mappings from the respective security and access control concerns. The iMM model of the system provides the particular values for elements (e.g., the security actions to perform). In this way, different concerns models, independently managed (maintained, evolved, mapped to target platforms, etc.) may be re-used for the implementation of different systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans  xmlns="http://www.springframework.org/schema/beans"
        xmlns:jaxws="http://cxf.apache.org/jaxws" ...>
        <!-- SERVICES - Warehouse A-->
        <jaxws:endpoint id="warehouseA" address="/warehouseA"
                        implementorClass="org.ws_i.sampleapplications....WarehouseAService">
                <jaxws:implementor>
                        <ref bean="warehouseAService"/>
                </jaxws:implementor>
                <jaxws:outInterceptors>
                   <ref bean="WarehouseEndTimestampSign_Response"/>
                </jaxws:outInterceptors>
                <jaxws:inInterceptors>
                   ...
                </jaxws:inInterceptors>
        </jaxws:endpoint>
        ...
        <bean id="warehouseAService" class="org.ws_i.sampleapplications....WarehouseAService">
                <property name="logStub" ref="loggingClient"/>
                <property name="manufacturerAService" ref="manufacturerAClient"/>
                ...
        </bean>
        ...
        <!-- WSS4JOutInterceptor for timestamping and signing the SOAP response. -->
        <bean
             id="WarehouseEndTimestampSign_Response"
             class="org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor">
             <constructor-arg>
                <map>
                   <entry key="action" value="Timestamp Signature"/>
                   <entry key="user" value="warehouse"/>
                   <entry key="signaturePropFile" value="warehouseKeystore.properties"/>
                   <entry key="passwordCallbackClass" value="org.ws_i.sampleapplications.
                                                          ...WarehousePasswordCallback"/>
                   <entry key="signatureParts" value="
                           {Element}{http://docs.oasis-open.org/wss/2004/01/
                                     oasis-200401-wss-wssecurity-utility-1.0.xsd}Timestamp;

                           {Element}{http://schemas.xmlsoap.org/soap/envelope/}Body"/>
                </map>
             </constructor-arg>
        </bean>
        ...
</beans>
```

Fig. 4. *Warehouse Service's Security Configuration*

4 Related Work

Among the proposals that deal with the different aspects supported by WS-Policy, the approach by Ortiz and Hernández [16] is one of the most representatives. Their solution makes use of aspect-orientation, Service Component Architecture (SCA) modeling and a UML (Unified Modeling Language) profile. Ours, in change, proposes the use of domain specific languages (DSLs), more adapted to each of the extra-functional aspects in hand, to later combine these DSLs into one integrated model. No means to generate the implementation of the security aspects previously mentioned are described in [16].

In [12], the use of a security view (a view where security properties are associated to functional elements in a process) is suggested. A first approximation towards the modeling of security characteristics in event-driven process chains (EPCs) is presented, modeling security at a technical level. Sharing the idea

of the use of views or different models for addressing NF concerns, both approaches differ (among other things) in that [12]'s doesn't make use of MDA's abstract-to-specific chain of transformations, not benefiting from MDA's reutilization capabilities, and that our approach isolates the process modeler from technical security knowledge.

A methodology for an end to end SOA security configuration in proposed by [18]. The approach is model-driven, and makes use of templates to express identified security patterns, initially added as abstract keywords that represent security requirements at business level, and then at a service model level. This mapping from business-level requirements to technical intents is a manual task, performed by a software architect. Security intents are complemented with a Security Infrastructure Model (SIM) to later generate a concrete configuration. The SIM, although closer to topology modeling, fulfills an equivalent functionality to that of the WS-SecurityPolicy platform model in this work. Our approach and the one in [18] differ in that the latter is limited to security, and in, perhaps, the greatest strength in this proposal: the use of separate concern models to address each non-functional characteristic, a feature that aids reuse and modularity of design and implementation.

A security metamodel is presented in [13] to model security considerations in business processes, map them to service-oriented architectures, and generate WS-SecurityPolicy configurations. The focus is on confidentiality, integrity, identification, and authentication using patterns. Security goals (primitives) are expressed in these models, and policy assertions satisfying these goals are specified. In our work, the use of domain specific independent metamodels to describe the individual concerns facilitates the understanding, analysis, and generation of the different aspects of the service individually. The work in [13] uses one unique metamodel to express all characteristics of the services, hindering, therefore, the autonomous evolution of the different concerns. Additionally, our work is not limited to security concerns, being flexible enough to integrate other type of policies. Nonetheless, the work in [13] is sound, and worth of consideration.

5 Conclusions and Future Work

This paper has presented a piece of work focused on the achievement of an integrated, model-driven solution for the development of service-oriented architectures with security and access control support. The different participant metamodels have been presented, and a well-known use case, mapped to an unmodified, state-of-the-practice commercial framework has been included to illustrate the usefulness of the approach.

Multiple domain-specific models independently addressing NF concerns, and being independently mapped into target platforms, foster concern models reuse, and favors that each concern be addressed in a convenient manner. Additionally, the modularity of the approach permits that each concern design, functional or not, is not polluted with extraneous information. Finally, the integrated model offers a practical way of measuring and analyzing the system in early stages of the development.

The proposed solution presents great flexibility in its evolution regarding the appearance of new standards or technologies. Moreover, the WS-Policy framework itself has been defined in a generic fashion, allowing for many standards to be formulated based on it. Consequently, all standards already available, or in process of being defined under its umbrella may easily be supported by this solution, and its assertions can readily be applied. This leverages its use for developing not only security-aware services, but also include reliability, timing constraints, secure exchange, transactions, etc.

With respect to future work, in the short term it will be focused on the completion of a prototype tool for the CIM-PIM-PSM chain. In a longer term, it is planned to consider other policy aspects under standardization process, to be able to use the information intrinsic to the assertions (for shaping of code generation, configuration of target platform, generation/use of an extra-functional aspect, etc.).

On a different token, the shifting from a generation approach based on a monolithic metamodel, towards a composition-based generation approach (in which any metamodel could be used to generate artifacts based on a set of weaving associations) could provide an alternative line of research.

Acknowledgment. This work was supported in part by the Centro Español de Desarrollo Tecnológico (CDTI, Ministry of Industry, Commerce and Turisms), by means of the ITECBAN (Infraestructura Tecnológica y Metodológica de Soporte para un Core Bancario) project, from the INGENIO 2010 program.

References

1. Apache. Apache CXF (2010)
2. Berre, A.: Service oriented architecture Modeling Language (SoaML)-Specification for the UML Profile and Metamodel for Services, UPMS (2008)
3. CDTI. ITECBAN
4. Dodd, J., Allen, P., Butler, J., Olding, S., Veryard, R., Wilkes, L.: CBDI-SAE Meta Model for SOA Version 2. Technical report, Everware-CBDI (2007)
5. Elrad, T., Aldawud, O., Bader, A.: Aspect-Oriented Modeling: Bridging the Gap between Implementation and Design. In: Batory, D., Consel, C., Taha, W. (eds.) GPCE 2002. LNCS, vol. 2487, pp. 189–201. Springer, Heidelberg (2002)
6. Del Fabro, M.D., Bézivin, J., Jouault, F.: AMW: a generic model weaver. In: Proceedings of the Using Metamodels to Support MDD Workshop, 10th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2005 (2005)
7. Finkelsetin, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke, M.: Viewpoints: A framework for integrating multiple perspectives in system development. International Journal of Software Engineering and Knowledge Engineering 2 (1992)
8. Kelly, S., Tolvanen, J.-P.: Domain-specific modeling: enabling full code generation. Wiley-IEEE, Hoboken, New Jersey (2008)
9. Kim, A., Luo, J., Kang, M.: Security Ontology to Facilitate Web Service Description and Discovery. In: Spaccapietra, S., Atzeni, P., Fages, F., Hacid, M.-S., Kifer, M., Mylopoulos, J., Pernici, B., Shvaiko, P., Trujillo, J., Zaihrayeu, I. (eds.) Journal on Data Semantics IX. LNCS, vol. 4601, pp. 167–195. Springer, Heidelberg (2007)

10. Larrucea, X., Alonso, R.: Modelling and Deploying Security Policies. In: WEBIST 2009 - Proceedings of the Fifth International Conference on Web Information Systems and Technologies, Lisboa, Portugal, pp. 411–414. INSTICC Press (2009)
11. Lodderstedt, T., Basin, D., Doser, J.: SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 426–441. Springer, Heidelberg (2002)
12. Jensen, M., Feja, S.: A Security Modeling Approach for Web-Service-Based Business Processes. In: 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, ECBS 2009, San Francisco, California, USA, pp. 340–347. IEEE Computer Society (2009)
13. Menzel, M., Meinel, C.: A Security Meta-model for Service-Oriented Architectures. In: 2009 IEEE International Conference on Services Computing, Bangalore, India, pp. 251–259. IEEE (September 2009)
14. Miller, J., Mukerji, J.: MDA Guide Version 1.0.1 (2003)
15. Mouelhi, T., Fleurey, F., Baudry, B., Le Traon, Y.: A Model-Based Framework for Security Policy Specification, Deployment and Testing. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 537–552. Springer, Heidelberg (2008)
16. Ortiz, G., Hernández, J.: Service-Oriented Model-Driven Development: Filling the Extra-Functional Property Gap. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 471–476. Springer, Heidelberg (2006)
17. Rashid, A., Sawyer, P., Moreira, A., Araújo, J.: Early Aspects: A Model for Aspect-Oriented Requirements Engineering. In: IEEE International Conference on Requirements Engineering, p. 199 (2002)
18. Satoh, F., Nakamura, Y., Mukhi, N., Tatsubori, M., Ono, K.: Methodology and Tools for End-to-End SOA Security Configurations. In: 2008 IEEE Congress on Services, SERVICES I, Honolulu, Hawaii, USA, pp. 307–314. IEEE Computer Society (2008)
19. Gallino, J.P.S., de Miguel, M.A., Briones, J.F., Alonso, A.: Model-Driven Development of a Web Service-Oriented Architecture and Security Policies. In: 2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, Carmona, Spain, pp. 92–96. IEEE Computer Society, Los Alamitos (2010)
20. SpringSource. Spring Framework (2010)
21. Sutton Jr., S.M.: N degrees of separation: multi-dimensional separation of concerns. In: International Conference on Software Engineering, pp. 107–119 (1999)
22. The Object Management Group (OMG). UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms Version 1.1 (2008)
23. Web Services Interoperability Organization, `http://www.ws-i.org`
24. WS-I. Sample Architecture Usage Scenarios (2003)
25. WS-I. Supply Chain Management Sample Architecture (2003)
26. WS-I. Sample Applications Security Architecture Document (2006)