

# An Aspect-Oriented Approach to Enforce Security Properties in Business Processes

Inaya Yahya<sup>1</sup>, Sameh Hbaieb Turki<sup>1</sup>, Anis Charfi<sup>2</sup>, Slim Kallel<sup>3</sup>, and Rafik Bouaziz<sup>1</sup>

<sup>1</sup>MIRACL, University of Sfax, Tunisia  
inaya.yahya@gmail.com, turkisameh@yahoo.fr  
raf.bouaziz@fsegs.rnu.tn

<sup>2</sup>SAP Research, Darmstadt, Germany  
first.lastname@sap.com

<sup>3</sup>ReDCAD, University of Sfax, Tunisia  
slim.kallel@fsegs.rnu.tn

**Abstract.** Security is an essential requirement for business processes. However, we observe that security is mostly addressed at the technical implementation level and not at the design level. In a previous work we motivated the need to address security already in business process modeling. In this paper, we show how one could use Aspect-Oriented Programming (AOP) to enforce security requirements in a modular way. Starting from a business process model where security requirements are expressed using a profile mechanism we generate AspectJ [1] code, which enforces those requirements. This generation is based on a set of Model-to-Text transformation rules. As security is a typical example for crosscutting concerns the usage of aspects allows for a modular implementation, in which the implementation of the business process is separated from the implementation of the security properties.

**Keywords:** AOP, security, Web services, Separation of concerns, MDA.

## 1 Introduction

The development of composite web services is a complex task, as it is based on technical and low-level languages such as WS-BPEL or programming languages. It requires a high expertise and can therefore not be done by non-technical users. Further, if a composite web services has to be implemented for different target platforms the implementation has to start from scratch each time due to the lack of reusable design models of the composite service. On the other hand, there are many works that use the model driven architecture (MDA) and model-driven software development to address such problems by raising the level of abstraction and fostering reuse through using design models. In a previous work [2], we presented a model-driven approach to composite web service development, which starts with modeling composite services using business process models defined in the Business Process Modeling Notation (BPMN) [3]. Then, these models are enriched with service related details using BPMN4SOA, which is a service-oriented extension to BPMN that we proposed in [6]. From these models we generate executable service composition code in WS-BPEL or

in java using appropriate transformations and code generators. An Eclipse based tool-set was developed to support our approach. In [4] we extended our approach to cover not only the functional side of service composition but also non-functional aspects such as security, quality of service, etc. In that work, we proposed a profile mechanism for BPMN in a similar way to the profile mechanism of UML. Based on this several profiles can be defined to express non-functional properties in BPMN and in BPMN4SOA. In [4] we also presented a security profile for BPMN.

In this paper, we extend the scope of our coverage for non-functional concerns by supporting the transition from modeling to implementation. In particular, we focus on mapping the security properties that can be expressed using the security profile to aspect code in AspectJ [1]. In addition, we present an Xpand [5] based code generation tool which implements that mapping and produces aspect code for enforcing the security properties at runtime. This new generator complements the code generator from BPMN4SOA to Java, which we implemented in a previous work [6]. With both generators we support the functional and non-functional aspects of service composition.

The remainder of this paper is organized as follows. Section 2 gives an overview of our previous work: we start by presenting a generic meta-model to express non-functional concerns in business processes, then we present the security profile which allows expressing security properties in BPMN process models. In Section 3 we present the mapping rules from the security profile to aspect code. In addition, we report on the code generator, which we built based on that mapping. In Section 4 we illustrate our proposal by an example. Section 5 discusses related works and Section 6 concludes this paper.

## 2 Background

In this section, we introduce the profile mechanism and the security profile, which we proposed in [4]. The non-functional profile that we proposed allows expressing QoS properties in a simple way. The business developer doesn't need to knowledge technical details relied to QoS properties; he just uses some annotations and specifies the value of their attributes. In opposed to some non-functional profiles which are proposed in the literature and expresses more technical concepts. Understanding those concepts is necessary for this paper as the security profile is the source for our mapping and the respective model-to-text transformation. We start by presenting the proposed profile mechanism and the underlying meta-model for expressing non-functional profiles. Then we give an overview of the security profile.

### 2.1 Meta-model for Non-functional Profiles

In analogy to the profile concept in the Unified Modeling Language (UML) [7] we propose a profile concept for process modeling languages such as BPMN [3]. In Figure 1 we present a meta-model for non-functional profiles and we detail the concepts of this meta-model in the following.

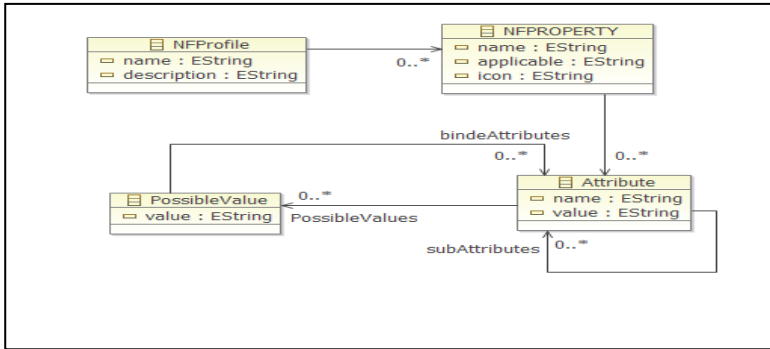


Fig. 1. The meta-model of non-functional profiles

**NfProfile:** represents a given concern that needs to be expressed in business processes such as security or business performance. It is characterised by two attributes “name” and “description”.

**NfProperty:** represents a property that belongs to a concern. Response time and cost are examples for such properties. It is characterised by three attributes “name”, “applicable” i.e. applicability constraints which specify for instance to which process elements a given non-functional property can be applied, and “icon” which specifies the graphical icon for each non-functional property.

**Attribute:** a property has zero or more attributes and each attribute has a name and a value. For example the property cost may have an attribute called metric and an attribute called amount. One Attribute may define several *subAttributes* which at the same time can be bound by one of the predefined available values of the parent meaning that they will only be available when the value that binds them is selected.

For applying our approach to BPMN and defining a BPMN profile for modeling non-functional concerns, we use the extension mechanism proposed in BPMN specification [3]. BPMN introduces the artifact concept to add non-standard elements. In our case, we defined two types of artifact: *NfProfile* and *NfProperty* as shown in Figure 2.

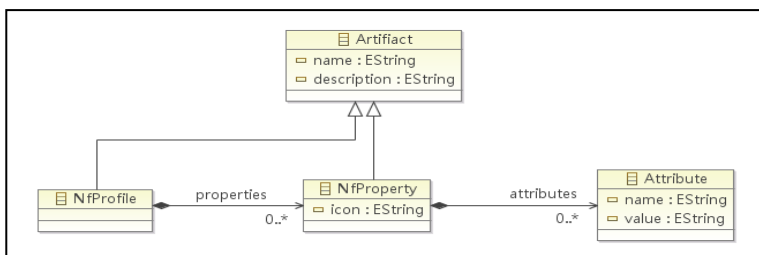







Fig. 2. Definition of the NfProfile and NfProperty artifacts according to BPMN extensibility [3]

## 2.2 Security Profile

As presented in [4], the security profile has been defined for providing a lightweight extension to BPMN for expressing security properties in business process model. This profile defines a set of annotations and their attributes. We note that we concentrated on the modeling and the implementation of the security aspects only side service composition process and not side partners.

In this paper we will concentrate only on five security properties as shown in the table below. Other properties can be added in this profile and mapped to aspects. To enforce security properties, we adopted a service-oriented approach, which allows invoking a set of web services to realize security aspects. For this reason, we defined for each security property two attributes: “Operation” and “Service” which the developer defines their values.

**Table 1.** Security properties

Security property	Additional Attributes	Notation
<b>Separation of Duties (SoD):</b> This property expresses that two tasks have to be performed by two different users or user roles to avoid the risk of frauds.	<ul style="list-style-type: none"> <li>• <b>Type:</b> The possible value are: Static SoD, Simple Dynamic SoD, Object-Based SoD, Operational SoD, Operational Object-Based SoD</li> </ul>	
<b>Binding of Duties (BoD):</b> This property expresses that two activities must be performed by the same user or by the same user role.	No one	
<b>Confidentiality:</b> This property expresses that data is confidential and should be only accessible to users with appropriate credentials. Confidentiality is ensured using an encryption algorithm	<ul style="list-style-type: none"> <li>• <b>Key:</b> the key used to ensure encryption/decryption operation.</li> </ul>	
<b>Integrity:</b> This property expresses that the data must not be modified by a malicious party when transmitted.	<ul style="list-style-type: none"> <li>• <b>Key:</b> the key used to ensure integrity operation.</li> </ul>	
<b>Authentication:</b> This property expresses that authentication is required for accessing some data object or for performing some activity.	No one	

## 3 Mapping the Security Profile to Enforcement Aspects in AspectJ

In this section, we first give an overview of AspectJ. Then, we present the mapping rules from security profile to AspectJ aspects.

### 3.1 Target Language for Enforcing Security Properties: AspectJ

To separate the functional and non-functional parts we use Aspect-Oriented Programming (AOP). The functional part of the composite web service, which is specified using the BPMN process model, is transformed into java code and the non-functional part is transformed into executable aspect code in AspectJ [1], which is an aspect-oriented extension to java. The weaving of these two executable codes is an automatic AspectJ task.

AspectJ provides the possibility to define aspects with their composition rules into the base java code. The main concepts in Aspect-Oriented Programming are join points (which are points in the execution of the program such as method calls), pointcuts (which allow to select one or more join points) and advices (which are behavioral units that contain the crosscutting logic similar to methods). The unit of modularization used by AspectJ is the aspect which is composed of one or more pointcuts and advices. Each pointcut is associated with an advice. AspectJ defines three types of advices: before advice, around advice and after advice. These advice are respectively executed before, instead, and after of the join point matched by their respective pointcuts. We show in the Listing 1 the structure of a logging aspect in AspectJ. The pointcut of this aspect selects all calls to public methods. A before advice is associated with this pointcut, which will print out a message before each matched join point.

```
public Aspect Logging_Aspect {
    pointcut pointcut_Logging(): call(public * *.*(..));
    before(): pointcut_Logging(){
        System.out.println (" call to method " +thisJoinPoint.toLongString());
    }
}
```

**Listing.1.** The structure of an aspect in AspectJ

With aspects our approach ensures that the security enforcement code is well modularized. The security expert needs to focus only on the security aspect to understand how security policies are enforced. He does not need to look at and understand the code implementing other concerns. Furthermore, when certain non-properties change, for example, an encryption module is upgraded or replaced, only the respective security aspects must be regenerated and redeployed. The core business processes and aspects enforcing other concerns remain unchanged.

### 3.2 Mapping Rules

In the following, we explain how each property in the security profile is mapped to aspect code in AspectJ and the corresponding java joint point in which this code will be weaved. We use the following conventions for presenting the following mapping rules:

- Assuming a service activity that calls a given operation. The respective pointcut of the enforcement advice will be named according to this pattern:

**PropertyName\_OperationName\_BPMNelement**

- The parameters of the generated pointcut correspond to either the input or the output value of the BPMN element in the business process model.
- We developed a java based library with helper functions which can be used to enforce the security properties and which can be called from the advice. For each security property, we call the respective enforcement function according to this naming pattern: *PropertyName\_Method()*.

• **The Confidentiality Property:** The confidentiality property can be applied to the messages. It is generally mapped to *before* advices. Only in the case of end message, it is mapped to an *after* advice. Figure 3 illustrates this mapping.

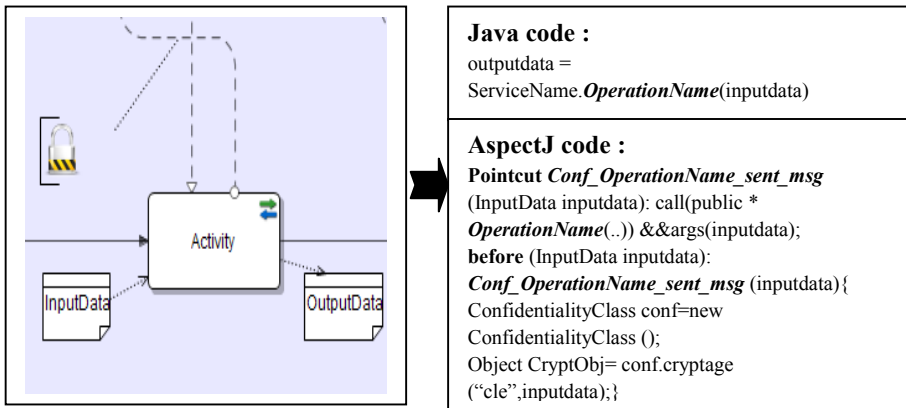


Fig. 3. Mapping of confidentiality

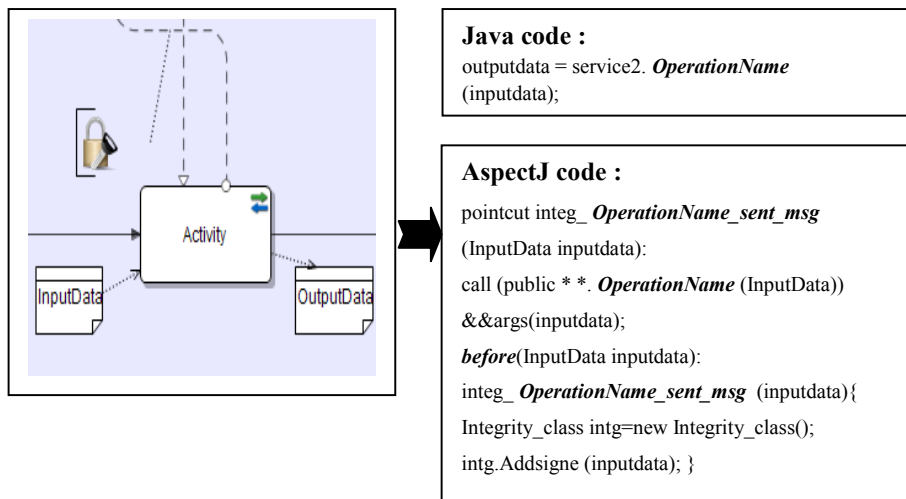


Fig. 4. Mapping of integrity

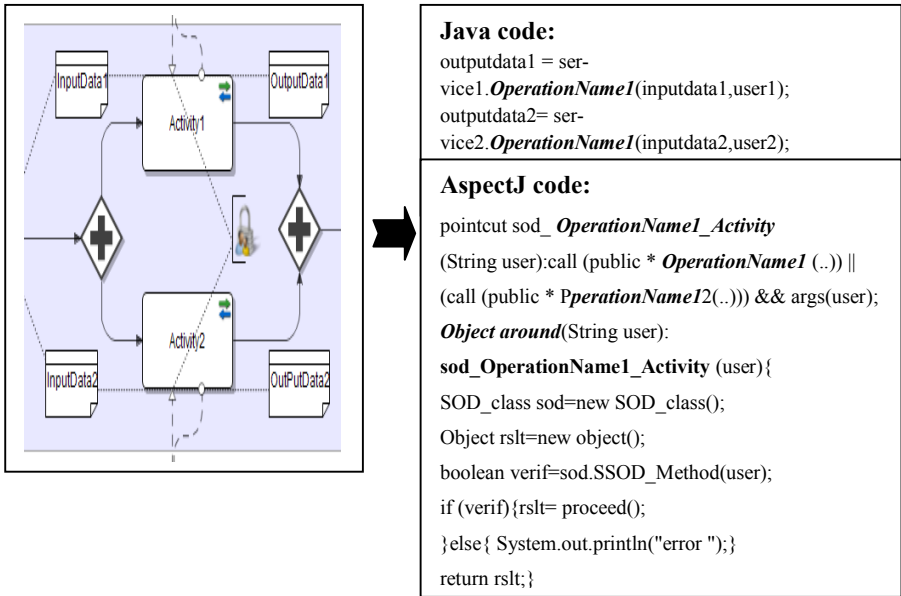


Fig. 5. Mapping of separation of duties

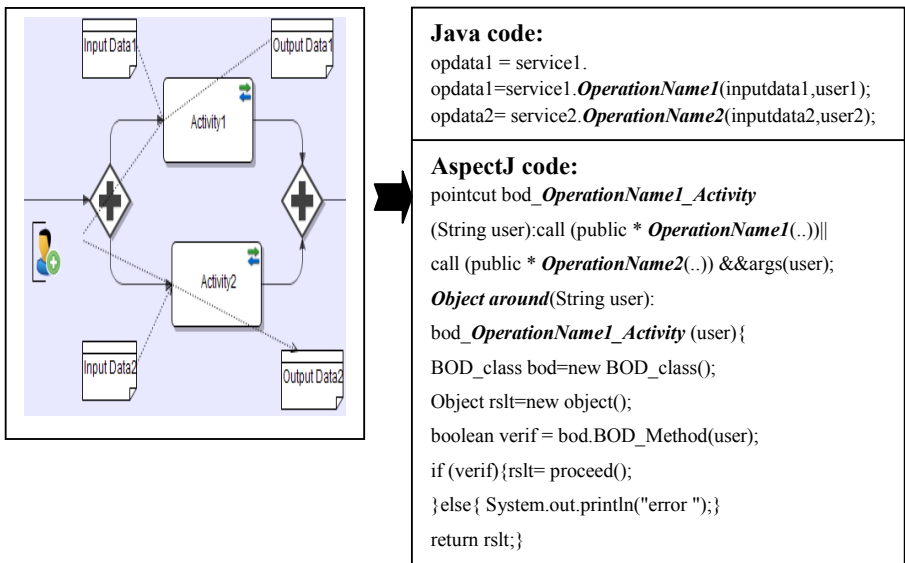


Fig. 6. Mapping of binding of duties

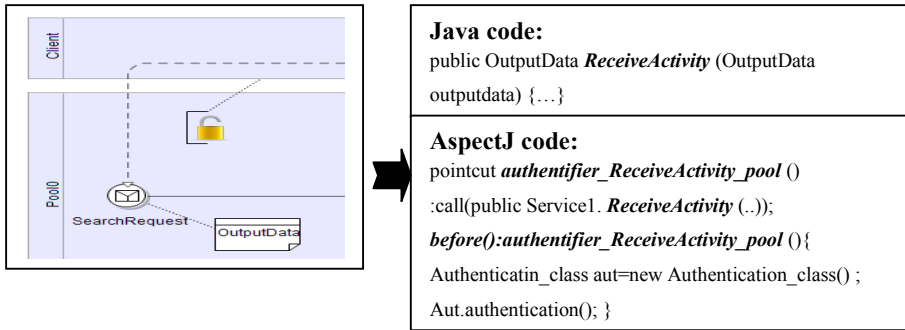


Fig. 7. Mapping of authentication

- **The Authentication property** is mapped to a *before* advice to guarantee the authentication of the client and the partner before interaction with the service composition pool. Figure 7 shows this mapping.
- **The Integrity Property:** The integrity property can be applied to the messages. It is generally mapped to a *before* advice. Only in the case of end message, it is mapped to an *after* advice, as shown in Figure 4.
- **The Separation of Duties Property (SOD)** is mapped to an *around* advice to guarantee the respect of this property by all activities of the group. Figure 5 shows this transformation.
- **The Binding of Duties Property:** The binding of duties (BOD) property is mapped to an *around* advice, which guarantees the respect of this property by all activities of the group. Figure 6 illustrates this mapping.

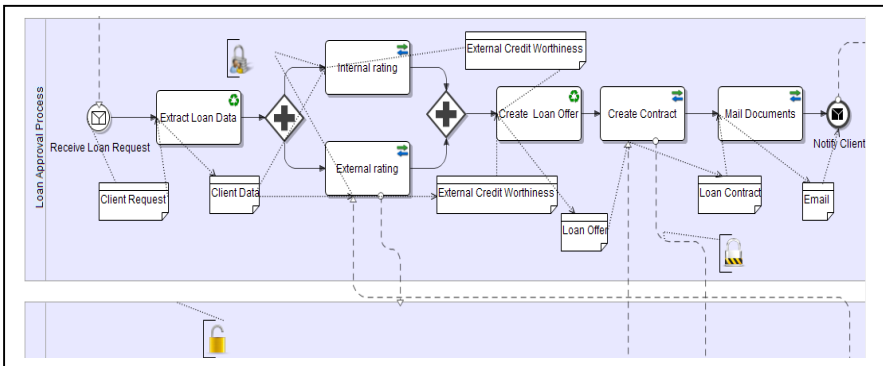
### 3.3 The AspectJ Code Generator

Based on the mapping presented above, we implemented a code generator from the security profile to AspectJ [1]. This generator is based on the Xpand [5] model-to-text transformation language. Based on an Ecore meta-model of the security profile and templates in Xpand implementing the mapping, the generator takes a business process model annotated by security properties and generates one or more text files with the AspectJ code. The AspectJ generator was developed as Eclipse plugin. For the definition of the source model we used the non-functional editor with annotations as shown in Figure 8. In a previous work [6], we implemented a BPMN to java generator, which is responsible for generating the functional part of the composite service. Both generators can be used together with ours to have both the functional code in java classes and the non-functional code in AspectJ aspects.



## 4 Case Study

To illustrate our approach we introduce the example of loan approval process (LAP) as shown in Figure 8. This process starts when the client applies for a loan from his bank. The bank will then execute two parallel rating activities. The first one ensures external rating using a credit-reporting agency. The second activity is for internal rating and it verifies the creditworthiness of the client based on the provided documents. If the evaluation of the customer creditworthiness is positive, an offer will be created and subsequently the contract documents will be generated using a contracting web service. Finally the offer and the contract are sent to the customer by email using an appropriate mailing web service.



**Fig. 8.** The Loan Approval Process (LAP) annotated with security properties

In this process we have sensitive data that is transmitted between the different services. Several security requirements arise in this scenario, which can be expressed using the security profile. For example, an authentication property is applied to the partner that will ensure external rating, as we see in Figure 8. That property is associated to the corresponding pool. A separation of duties property is used to express separation of duties between the two parallel rating activities. In fact, these activities should be performed by two different users to avoid the risk of fraud. The message sent to the partner “contracting service” contains confidential information and therefore needs to be encrypted before being sent. To specify this requirement we linked the confidentiality property to that message

Figure 8 shows the LAP process model defined in BPMN and annotated with the properties of the security profile. After modeling the process and annotating it with properties of the security profile we use our AspectJ generator to generate security aspects that enforce the defined security properties. The java generator can be used to generate java classes that implement the modeled composite service.

The generated code is an executable and complete AspectJ code that is organized in three aspects for this example: an aspect for each security property. The first

property is authentication, which is enforced using an advice that authenticates the partner before starting the process. As the process is implemented with a generated method called main, the pointcut of the advice matched that method as we see in Listing 2. The second property in this example is separation of duties and the corresponding aspect code is shown in Listing 3. Here, all activities of the group will be captured as join points and we need to guarantee that their respective users are different. The third property in this example is confidentiality and the corresponding aspect is shown in the Listing 4. The advice encrypts the message of the partner invocation before it is sent. This invocation is captured as join point by the pointcut associated with the generated advice. Regarding the advice, we have chosen to implement a java library with methods that enforce each security properties. The advice simply calls those methods. The least step is to combine these aspects with the functional java code to obtain an executable application.

```
public aspect AuthenticationAspect{
    pointcut Authentication_External Rating():execution(public void main(..));
    before():Authentication_External Rating(){
        Authentication_class authentication_class = new Authentication_class ();
        authentication_class.AuthenticationMethod(); } }
```

**Listing. 2.** The Authentication aspect

```
public aspect SOD_Aspect{
    pointcut SOD_Externalrating_Activity(ClientData newClientData , String user):(call(public *
    *.getInternalRating(...String)) || call(public * *.getExternalRating(...String)))&&
    args(newClientData,user)&&!within(LoanApprovalProcessAspect);
    Object around(ClientData newClientData , String user):
    SOD_Externalrating_Activity(newClientData,user){
        SOD_class sOD_class = new SOD_class ();
        boolean exist= sOD_class.SOD_Method("SSOD",newClientData,user);
        Object rslt=new Object();
        if (exist){c= proceed(newClientData,user); }else{ System.out.println("error ");}
    return c;}}
```

**Listing. 3.** The Separation of duties aspect

```
public aspect Confidentiality_Aspect{
    pointcut Confidentiality_CreateContract_outgoingMessage(LoanOffer newLoanOffer,String user)
    :(call(public * *.getLoanContract(..)) &args(newLoanOffer,user)&&!within(Confidentiality_Aspect));
    before(LoanOffer newLoanOffer,String user): Confidentiali-
    ty_CreateContract_outgoingMessage(newLoanOffer,user){
        newLoanOffer.setClientId(Confidentiality_class.Encryption(0001,newLoanOffer.getIdClient()));
        newLoanOffer.setamount(Confidentiality_class.Encryption(0001,newLoanOffer.getamount()));
        newLoanOffer.setdata(Confidentiality_class.Encryption(0001,newLoanOffer.getdata()));}
```

**Listing. 4.** The Confidentiality aspect

## 5 Related Work

The model-driven security in the context of SOA is an emerging research area.

In [8], the authors describe a model-driven architecture that allows the generation of web service security configurations from an UML model. The business process model and security intentions are modeled using standard UML diagrams. According to the MDA approach, the users define the application model enriched by the security intentions, and then detailed security configurations are generated. That work is based on transformations over UML constructs and a security environment model. Unlike our work, the approach presented in [8] does not support composite web services. It also does not support java code generation.

In [9] the authors define an approach, which ensures the generation of security configurations from business process models. In the first step, the business process model is annotated with abstract security intents using different models and notations: like UML'S SOAML profile, UML's QoS profile, and secureUML. The second step consists in composing functional models with access control models or security models. The final step consists in the code generation for specific target platforms. The main limitation of this work is the necessity composing different models, which requires different weaving associations and composition rules. In our work, we defined a common meta-model to support all non-functional properties and we can use different annotations from different profiles for the same business process model.

In [10], the authors propose a model-driven approach that facilitates the transformation of architecture models annotated with simple security intentions to security policies. This transformation is driven by security configuration patterns. The authors propose also a concise domain specific language (DSL) for expressing their security configuration patterns. A pattern engine is provided to execute security intentions and provide corresponding solutions. Comparing with our approach, this work is specific to security aspects and requires the extension of the domain specific language to support others non-functional concerns. Our approach is more generic and support different concerns without requiring any extension. The user can easily define a new non-functional profile and apply this to the business process model.

In [11], the authors propose a methodology for end-to-end security configuration for SOA applications and tools for generating security configurations from the requirements specified in previous phases of their approach. It makes it possible to configure security properly without increasing the workload of the developers.

In [12], the authors propose an extension of BPMN to ensure modeling security requirements into business process models. This extension allow user to incorporate seven security requirements represented by the same symbol (padlock) and for each security requirement a specific capital letter is added on the center of the symbol. The main limitation of this work is that do not provide an approach to generate executable code from the model.

The major advantage of our work over the works mentioned above is that our approach is generic and can be applied to any non-functional concern. It is not specific to security only. A further advantage is the possibility to use multiple non-functional profiles together. We focused in this paper only on security properties mapping, but others mappings can be defined for others QoS properties. In addition, we generate a modular code, which implements non-functional concerns.

## 6 Conclusion

In this paper, we presented an aspect-oriented approach to mapping the security properties expressed with a profile mechanism for BPMN to modular enforcement code in AspectJ. We implemented also an AspectJ code generator, which is based on the Xpand transformation language.

As future work, we aim to define others mapping rules from others non-functional profiles such as temporal properties to aspects. Another direction is to generate aspects in other AOP languages such as AO4BPEL for enforcing non-functional properties within BPEL process.

## References

1. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.G.: An Overview of AspectJ. In: Lindskov Knudsen, J. (ed.) ECOOP 2001. LNCS, vol. 2072, pp. 327–353. Springer, Heidelberg (2001)
2. Charfi, A., Turki, S.H., Chaâbane, A., Bouaziz, R.: A model-driven approach to developing web service compositions based on BPMN4SOA. *J. Reasoning-Based Intelligent Systems* 3(3/4) (2011)
3. Object Management Group: Business Process Modeling Notation (BPMN) 2.0, <http://www.omg.org/spec/BPMN/2.0>
4. Turki, S.H., Bellaaj, F., Charfi, A., Bouaziz, R.: Modeling Security Requirements in Service Based Business Processes. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Wrycza, S. (eds.) BPMDS 2012 and EMMSAD 2012. LNBP, vol. 113, pp. 76–90. Springer, Heidelberg (2012)
5. Eclipse Xpand Project, <http://www.eclipse.org/modeling/m2t/?project=xpand>
6. Chaâbane, A., Turki, S.H., Charfi, A., Bouaziz, R.: From Platform Independent Service Composition Models in BPMN4SOA to Executable Service Compositions. In: Proc. of iiWAS, France, pp. 653–656 (2010)
7. OMG.: UML: Superstructure version 2.0 (2005), <http://www.omg.org/spec/UML/2.0/>
8. Nakamura, Y., Tsubori, M., Imamura, T., Ono, K.: Model-driven security based on Web services security architecture. In: Proc. of SCC, Florida, USA, pp. 7–15 (2005)
9. Gallino, J.P.S., Miguel, M., Briones, J.F., Alejandro, A.: Domain-Specific Multi-Modeling of Security Concerns in Service-Oriented Architectures. In: Proc. of SCC, Washington, USA, pp. 761–762 (2011)
10. Menzel, M., Warschofsky, R., Meinel, C.: A Pattern-driven Generation of Security Policies for Service-oriented Architectures. In: Proc. of ICWS, Florida, USA, pp. 243–250 (2010)
11. Satoh, F., Nakamura, Y., Mukhi, K.N., Tsubori, M., Ono, K.: Model-Driven Approach for End-to-End SOA Security Configurations. In: Non-Functional Properties in Service Oriented Architecture: Requirements, Models and Methods, ch. 12, pp. 269–298 (2011)
12. Rodriguez, A., Piattini, E.F.-M.M.: A BPMN Extension for the Modeling of Security Requirements in Business Processes. *J. IEICE - Transactions on Information and Systems* E90-D(4), 745–752 (2007)