Service-Oriented Model-Driven Development: Filling the Extra-Functional Property Gap*

Guadalupe Ortiz and Juan Hernández

Quercus Software Engineering Group
University of Extremadura
Computer Science Department
Spain
{gobellot, juanher}@unex.es

Abstract. Although vendors provide multiple platforms for service implementation, developers demands approaches for managing service-oriented applications at all stages of development. In this sense, approaches such as *Model-Driven Development* (MDD) and *Service Component Architecture* (SCA) can be used in conjunction for modeling and integrating services independently of the underlying platform technology. Besides, WS-Policy provides a XML-based standard description for extra-functional properties. In this paper we propose a cross-disciplinary approach, in which the aforementioned MDD, SCA and WS-Policy are assembled in order to develop extra-functional properties in web services from a platform independent model.

Keywords: Extra-Functional property, web service, model-driven development, aspect oriented techniques, WS-policy, service component architecture.

1 Introduction

Web Services provide a successful way to communicate distributed applications, in a platform independent and loosely coupled manner. Although development middlewares provide a splendid environment for service implementation, methodologies for earlier stages of development, are not provided in a cross-disciplinary scope. At present, academy and industry are beginning to focus on the modeling stage; two representative approaches are described below:

To start with, SCA provides a way to define interfaces independently of the final implementation [3]. This proposal allows the developer to define a high level model; however, it does not face how to integrate this definition with other stages of development. As the second trend, many proposals are emerging where MDA is being applied to service development. MDA solves the integration of the different stages of development, but it does not provide a specific way to do so for service technology.

Let us consider now that we want to provide our modeled services with extra-functional properties; the way to do so has not been considered yet neither by SCA approach nor by model-driven ones. On the other hand, WS-Policy provides a

* This work has been developed thanks to the support of MEC under contract TIN2005-09405-C02-02.

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 471 – 476, 2006.
© Springer-Verlag Berlin Heidelberg 2006

standardized way for describing extra-functional service capabilities; however, it does not determine how the properties are to be modeled or implemented.

Closely related to this is the aim of this paper, which consists on offering a model-driven methodology in order to deal with extra-functional properties, that is, additional functionality which should not be part of the system's main functionality.

The rest of the paper is organized as follows: *Section 2* gives an overview of the whole process followed in this approach. *Section 3* shows how the PIM should be implemented; then, *Section 4* explains the PSM stage. *Section 5* explains the rules used to obtain code from PSMs. Other related approaches are examined in *Section 6*, whereas the main conclusions are presented in *Section 7*.

2 Model-Driven Transformations

In this section we provide and depict (*Figure 1*) a general overview of the presented approach: The first thing to be done is to define the metamodel to be followed by the platform independent model. In this sense, we propose to use a UML profile (MOF compliant) for the platform independent model, consequently the UML metamodel is our PIM's metamodel. Afterwards, three different metamodels are proposed in this approach for PSM stage, which will be explained later on.

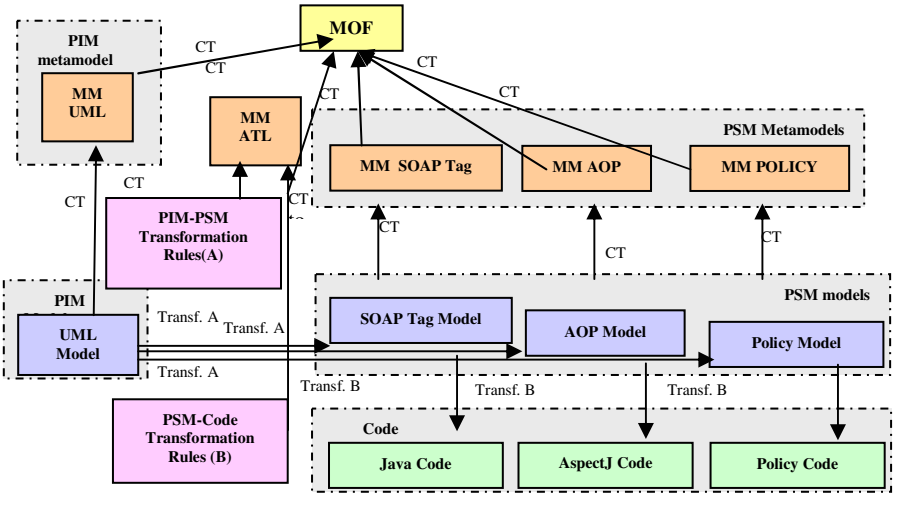

Fig. 1. ATL transformation process

Subsequently the transformation from the PIM to the PSMs has to be defined. We used *ATL* tool, which provides an Eclipse plugin and which has its own model transformation language. The ATL transformation file will define the correspondence between the elements in the source metamodel (PIM) and the target ones (PSMs) and will be used to generate the target model based on the defined rules and the input model. Finally, once we have obtained the specific models, we will transform them into code automatically by using additional transformation rules.

3 Extra-Functional Property PIM

In order to maintain our system loosely coupled when adding extra-functional properties to the model, we propose the profile in *Figure 2,* whose elements will be explained as follows and are deeply detailed in [9]:

- First of all, we define the stereotype *extra-functional property*, which extends *operation metaclass* or *interface metaclass*: when applied to an interface the property will be applied to all the operations in it. The stereotype provides five attributes: *actionType* indicates whether the property behavior will be performed *before*, *after* or *instead of* stereotyped operation's execution – or if no additional behavior is needed it will have the value *none,* only possible in the client side, which is out of the scope of this paper. Secondly, *optional* will indicate whether the property is performed optionally or compulsorily. *ack*, when *true* it means that it is a well-known property and its functionality code can be generated at a later stage; it will have the value *false* when it is a domain-specific property and only the skeleton can be generated. Finally, *PolicyId* will contain the name to be assigned to generated policy; *policyDoc* will be the url where the policy will be available.
- In order to define *actionType*, an *enumeration* is provided with four alternative values: *before*, *after, instead* or *none*.
- If the property is applied in an offered interface, then it will be implemented when the stereotyped operations are executed. If the property is applied in a required interface, it will be performed when the operations are invoked; in this case the service is acting as a client, which is beyond the purpose of this paper.

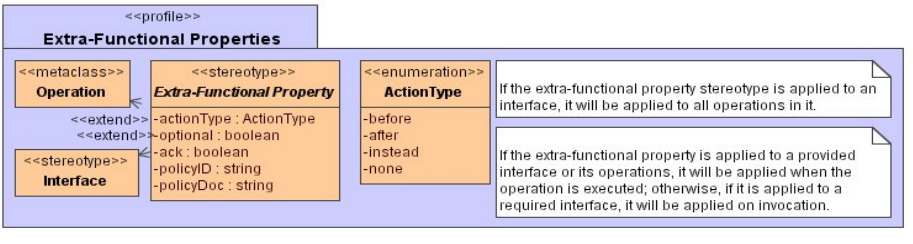

Fig. 2. Extra-functional property profile

Consider now we have a tourist information service. We will represent this service as a component, stereotyped as *serviceComponent*, which will offer a provided interface. Let us imagine now that we want to include some extra-functional properties to the *touristInformation* service model. We have devised three different sample properties:

- First of all, a *Log* property, which will be applied to all the operations offered by the service to record information related to all received invocations.
- Secondly, *RealTime* will be required discretionarily by the client when invoking *weatherInformation*: subject to a different pricing, the real time weather in a city may be obtained; under the regular price the average weather will be obtained.

- Finally, a *Login* property, used to control access to *RentingCar*, since it will only be possible to rent a car for those who have a username and a password.

We have to extend the extra-functional property stereotype with the specific properties to be applied, as shown in *Figure 3*. Properties are added then to *touristServiceInterface*. In order to provide all the operations with *Log*, we have stereotyped the interface with *log,* whose attributes indicate that the property will be performed *after* any operation in the interface is executed (not optional); information will be recorded in *logFile*; it is a well-known property; *policyID* is *Log* and *policyDoc* is null. *RealTime* and *Login* would also be interpreted similarly.

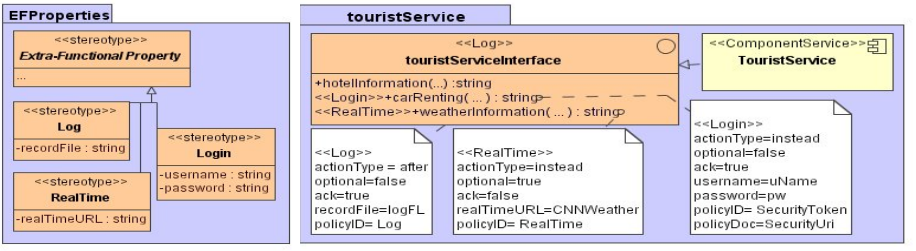

Fig. 3. PIM with extra-functional properties

4 Extra-Functional Property PSMs

Regarding target metamodels, there will be three of them: our specific models will be based, first of all, on an aspect-oriented approach to specify the property behavior, secondly on a *soap tags*-based approach, to lay down the necessary elements to be included or checked in the SOAP message header and, finally, a policy-based one for property description. Metamodels are explained and depicted below:

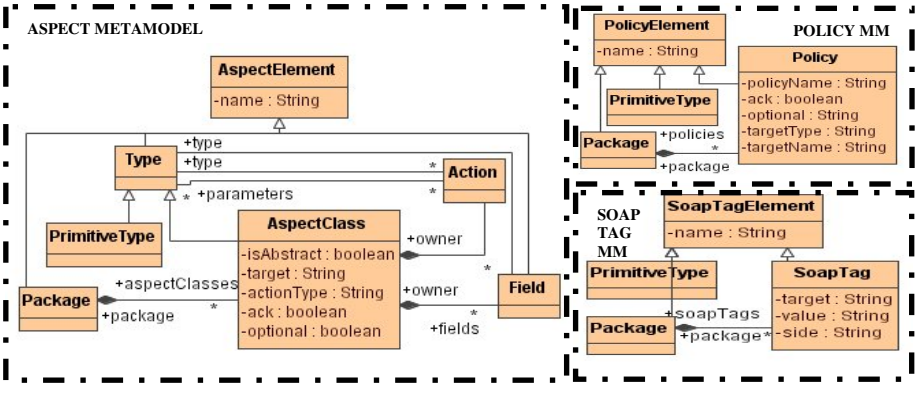

Fig. 4. PSM metamodels

- Every *aspectClass* will have an attribute *target* which indicates the method or interface for the property to be applied; *actionType* informs of when the property has to be applied; *ack* indicates whether the property is well-known or not and, finally, an *action* provides the corresponding functionality. All additional characteristics from the particular property will be included as attributes.
- New tags are included in the SOAP Header to select –client - or to check –service - the properties to be applied, when optional, or to deliver additional information [8]. Every Soap tag element will have a *target* which will instruct the method or interface for the property to be applied, *value,* to indicate the tag to be included; and *side*, which indicates whether it is included from the client or service side.
- A policy will be generated for every property. The policy will contain the policy *name*, whether the property is *optional* or not, well-known o domain-specific (*ack*); *targetType* indicates if the policy is to be applied to a *portType* or to an *operation* and *targetName* gives the name of the specific element.

Once the transformation rules are applied to the PIM, the specific models containing information about the properties are obtained, which are going to be described. Due to space restrictions only one characteristic element from each metamodel has been described: Log is applied to the interface, thus one aspect will be generated for each operation in the interface. The aspect would be named *TS_hotelInformation_Log*; *target* and *actionType* would have the values *touristServiceInterface.hotelInformation* and *after*, respectively, and *action* would be also included. Additionally, *ack* would have the *true* value. Regarding the policy element, *name* would be *http://aop4ws/Log*; *optional* would be *false*. Finally, for *policyAttachment, targetType* would take *portType* and *targetName touristServiceInterface*. The *RealTime* transformation would be similar, but being *optional*, a new attribute would be included in the SOAP Handler.

5 Code Layer: Extra-Functional Property Generated Code

In the last transformation stage, Java code will be generated to check if soap tags are included in the SOAP message and AspectJ code will be created for the aspects, , thus maintaining properties decoupled from the services implemented and the transparency in the property selection by the client [8]. With regard to description, it is proposed to generate the WS-Policy [1] documents for each property, which are integrated with the aspect-oriented generated properties [10]. This allows properties to remain decoupled not only at modeling stage, but also in during implementation.

ATL tools have also been used for code generation, where the transformation rules identify the different elements in the specific models and generate code from them. Transformation rules will generate skeleton code for the three model elements:

An AspectJ aspect will be generated for each aspect class in our model. Pointcuts will be determined by the execution of the target element. Regarding the advice, depending on the *actionType* attribute value, *before*, *after* or *instead*, the advice type will be *before*, *after* or *around*, respectively. Regarding property description, an xml file based on the WS-Policy standard is generated and attached to the service by the file based on the WS-PolicyAttachment standard. Code for the inclusion of SOAP Tags will be generated for every SOAP Tag Class in the model.

6 Related Work

As regards Web Service modeling proposals, the research presented by J. Bezivin et al. [4] is worth a special mention; in it Web Service modeling is covered in different levels, using *Java* and *JWSDP* implementations in the end. It is also worth mentioning the paper from M. Smith et al. [11], where a model-driven development is proposed for Grid Applications based on the use of Web Services. Our work provides the possibility of adding extra-functional properties to the services and is not oriented to the service modeling itself; thus it could be considered as complementary to them.

Concerning proposals which focus on extra-functional properties, we can especially mention the one from D. Fensel et al. [7], where extra-functional properties are modeled as pre and post conditions in an ontology description. Secondly, L. Baresi et al. extend WS-Policy by using a domain-independent assertion language in order to embed monitoring directives into policies [2]. Both are interesting proposals, however they do not follow the UML standard, which we consider essential for integrating properties in future service models.

7 Conclusions

This paper has shown a model-driven approach to including extra-functional capabilities in web service development in a loosely coupled manner. Properties are included in the PIM by using a UML profile. The initial PIM has been converted into three specific models which conform to three provided metamodels, based on the soap tags information, aspect oriented elements and policy based ones. All the code related to extra-functional property has been automatically generated from PSMs.

References

[1] Bajaj, S., Box, D., Chappeli, D., et al. Web Services Policy Framework (WS-Policy), ftp://www6.software.ibm.com/ software/developer/library/ws-policy.pdf, September 2004

[2] Baresi, L. Guinea, S. Plebani, P. WS-Policy for Service Monitoring. VLDB Workshop on Technologies for E-Services, Trondheim, Norway, September 2005

[3] Beisiegel, M., Blohm, H., Booz, D.,et al. Service Component Architecture. Building Systems using a Service Oriented Architecture. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-sca/SCA_White_Paper1_09.pdf, November 2005

[4] Bézivin, J., Hammoudi, S., Lopes, D. et al. An Experiment in Mapping Web Services to Implementation Platforms. N. R. I. o. Computers**:** 26, 2004

[5] Fensel, D., Bussler, C. The Web Service Modeling Framework WSMF. http:// informatik.uibk.ac.at/users/c70385/ wese/wsmf.bis2002.pdf

[6] Ortiz G., Hernández J., Clemente, P.J.How to Deal with Non-functional Properties in Web Service Development, Proc. Int. Conf. on Web Engineering, Sydney, Australia, July 2005

[7] Ortiz G., Hernández J., Toward UML Profiles for Web Services and their Extra-Functional Properties, Proc. Int. Conf. on Web Services, Chicago, EEUU, September 2006.

[8] Ortiz, G., Leymann, F. Combining WS-Policy and Aspect-Oriented Programming. Proc. of the Int. Conference on Internet and Web Applications and Services, Guadeloupe, French Caribbean, February 2006.

[9] Smith, M., Friese, T. Freisbelen, B. Model Driven Development of Service-Oriented Grid Applications. Proc. of the Int. Conference on Internet and Web Applications and Services, Guadeloupe, French Caribbean, February 2006