

# Weaving AspectJ aspects by means of transformations

A. M. Reina, J. Torres

Languages and Computer Systems Department  
University of Seville

Avda. Reina Mercedes, s/n. 41012, Seville. Spain  
{reinaqu, jtorres}@lsi.us.es

## Abstract

*In the last few years, new software paradigms, such as Aspect-Oriented Software Development (AOSD) or Model Driven Development (MDD), have been brought up in order to improve software adaptability to changes. MDA improves the adaption to different technologies by means of three different levels of modelling. This paper is focused on the platform specific level, and proposes the use of transformations to weave AspectJ aspects and the basic functionality at the modelling level before the code generation phase.*

## 1. Introduction

In the last few years software engineers have been worried about software evolution and how to face up to changes at a low cost. Software projects not only have to deal with the variability of requirements but also with changes of technology. Thus, new paradigms, such as the Aspect Oriented Software Development (AOSD [2]) or the Model Driven Development (MDD), have been brought up in order to improve software adaptability to changes. On the one hand, AOSD helps to face up to the variability of requirements because concerns are better localized. On the other hand, MDD, and specially, the Model Driven Architecture (MDA [5]) proposed by the OMG, improves the adaptation to new technologies due to its different levels of models: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM). Therefore, if we mix both paradigms, we will take advantage of the benefits of both of them, and that's the reason why we propose a hybrid approach, where weaving is made at modelling by means of transformations.

In the context of model driven development, transformations play a key role. According to [3], we can see a transformation as an instance of a relationship between one input domain and one output domain. Moreover, as metamodels can be seen as Abstract Syntax Graphs (ASG) for Domain

Specific Languages (DSLs), the transformation of a source model to produce a target model can be performed across a relationship among metamodels which describe the source and target models.

In [6] it was stated that current aspect modelling languages are platform-specific, in the sense, that they require the use of platforms that can handle concerns separately. Thus, the use of domain specific languages for modelling aspects at PIM level was proposed. In an ideal situation, the concerns specified separately at the PIM level should be maintained separated at PIMs, PSMs and code. But sometimes this is not possible, because the customer requires the use of a specific platform which is not aspect-oriented. In these situations, we propose the maintenance of the separation as far as possible.

In this paper we focus on the PSM level and propose a way to deal with non-aspect oriented platforms. We achieve this aim by means of the definition of a set of horizontal transformations to weave AspectJ aspects and Java classes.

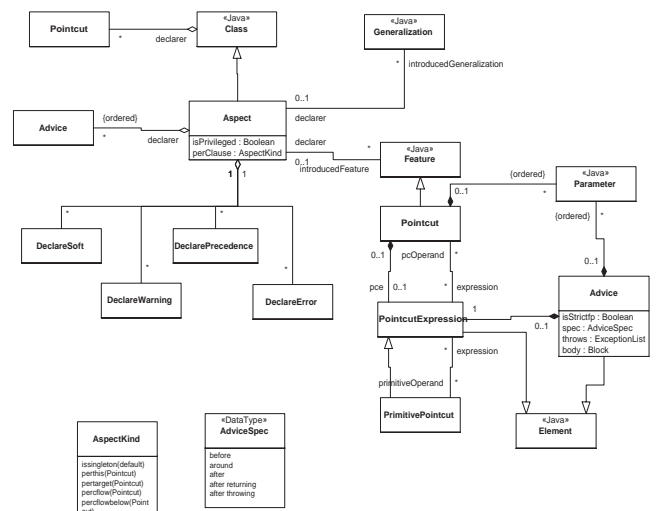


Figure 1. AspectJ Metamodel

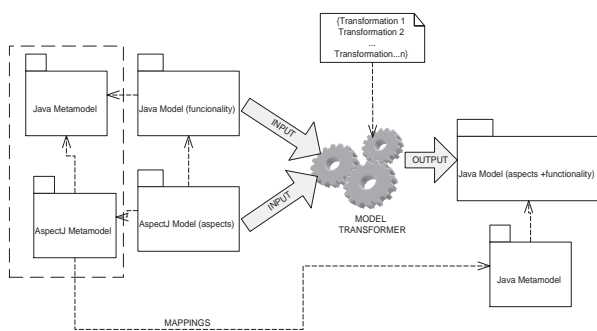
The rest of the paper is structured as follows: section 2 gives a general overview of our proposal and analyzes some of its advantages. Finally, section 3 concludes the paper and points out our future lines of research.

## 2. A brief overview

Although one of the main aims of the approach is the definition of a set of horizontal transformations to weave aspects at modelling, we focus in this section on the description of the approach and its advantages, due to the lack of space.

The elements we manage in our approach are: one source model (in which aspects and basic functionality are modelled separately), one target model (in which there is no separation of concerns), and a set of transformations. But, as we have pointed out previously, in order to implement the transformations we need to define a set of mappings between the metamodel of the source model and the metamodel of the target model. And, as a consequence, we also need to define the metamodels for the source and target models.

As ASG for the source model we have chosen the self-contained AspectJ metamodel defined in [4] (Figure 1), which extends the Java metamodel defined in [1]. This Java metamodel was chosen because is not an extension of the UML metamodel, it is a leaner version which tailors UML metaclasses to the Java 2 Specification. Having chosen this AspectJ metamodel, the target metamodel is totally determined: the same Java metamodel extended by the AspectJ metamodel. Figure 2 depicts a scheme of the different elements of our approach.



**Figure 2. Elements of the approach**

Another important element that should be defined is the language to express the mappings between metamodels. In this case, we have chosen the QVT draft [7], because it is the current response to the RFP. QVT stands for Query, Views and Transformations. Queries and transformations take models and perform actions upon them, while views

are models and are related to the model form which they have been created.

One of the main advantages of this approach is that we can change the platform in two different ways: firstly, we can define another set of mappings between the Java metamodel, and another metamodel, for example, a C++ metamodel; secondly, we could change the transformations and the target metamodel, and weave aspectJ aspects into a C++ model.

## 3. Conclusions and further work

We have defined an approach to weave AspectJ aspects and a Java classes at modelling. This let us produce not only Java code, but code written in any other language with just changing the set of transformations and the target metamodel. We also can change the written code defining a set of mappings between the Java metamodel and a metamodel defined for the target language.

As a future work, we want to continue obtaining a deeper knowledge of transformations and raising the level of abstraction, that is, defining domain specific languages for modelling aspects at the PIM level, and applying transformations to them.

## 4. Acknowledgments

This work has been partially supported by the *Spanish Ministry of Science and Technology* and FEDER funds: TIC-2003-369.

## References

- [1] S. Dedic and M. Matula. Metamodel for the java language. web.
- [2] T. Elrad, R. E. Filman, and A. Bader. Aspect-oriented programming. *Comm. ACM*, 44(10):29–32, Oct. 2001.
- [3] J. Greenfield and K. Short. *Software Factories. Assembling Applications with Patterns, Models, Frameworks and Tools*. Wiley Publishing, Inc., 2004.
- [4] Y. Han, G. Kniesel, and A. Cremers. Towards visual aspectj by a meta model and modeling notation. In *Proceedings of the 6th International Workshop on Aspect-Oriented Modeling held in conjunction with the 4th International Conference on Aspect-Oriented Software Development (AOSD'05)*, Mar 2005.
- [5] OMG. Mda guide version 1.0. Technical Report omg/2003-05-01, OMG, May 2003.
- [6] A. M. Reina, J. Torres, and M. Toro. Towards developing generic solutions with aspects. In *Proceeding of the Workshop in Aspect Oriented Modelling held in conjunction with the UML 2004 Conference*, oct 2004.
- [7] T. C. Services. Revised submission for mof 2.0 query / views / transformations rfp. version 1.1. Technical report, Tata Consultancy Services, aug 2003.