Multi-step Concern Transformation

Andrés Yie[1,2]*

[1] Grupo de Construcción de Software, Universidad de los Andes, Colombia
[2] System and Software Engineering Lab (SSEL), Vrije Universiteit Brussel, Belgium
ayiegarz@vub.ac.be

Abstract. A Model-Driven Software Product Line (MD-SPL) uses metamodels, models, and transformations to create a family of products. However, sometimes an MD-SPL must evolve to introduce new non-functional requirements, such as security or logging, in its generated products. Since the transformations used in a MD-SPL are fragile and complex, it is necessary to avoid changing them. We propose a non-invasive approach to add new non-functional requirements to the generated products, while keeping the original MD-SPL's transformations unchanged.

1 Model Transformation Lines Background

A Software Product Line (SPL) exploits the common characteristics of a product family. These common characteristics are mapped to a common architecture, components, and assets to assemble individual products. An SPL can be implemented using the Model-Driven Engineering approach. A Model-Driven Software Product Line (MD-SPL) [1] uses metamodels, models, and transformations as assets to generate a wide family of products.

In an MD-SPL, a high-level model describes a specific member of the product family using concepts from the specific domain (business) and this model does not contain any technological platform concepts. Usually a business specialist is in charge of building this model. Transformations are used to transform this model into a low-level model and finally to translate them into application code. Moreover, the use of platform independent models enables an MD-SPL to target several technological platforms starting from the same application specification.

An MD-SPL is built using a Model Transformation Line (MTL) [2] to generate an application from a high-level model. To cope with the complexity of transforming a high-level model into application code, an MTL performs several small, consecutive transformation steps. At each step, a set of transformation rules is executed trying to tackle a specific interest at time. For instance, the software development stages (business analysis, architecture definition, design, and implementation) can be used to generate an application from a high-level model. Figure 1 presents an MTL where a high-level model is the input for the MTL and at each step, transformations generate a new model using lower-level concepts.

* This research was supported by the Flemish Interuniversity Council (VLIR) funded CARAMELOS project (http://ssel.vub.ac.be/caramelos/)

For example, a model expressed using business concepts is transformed into an architectural model with concepts such as layers and communications among them. Next, a new transformation is applied an it generates a design model with specific platform concepts in it, such as the EJB's used to implement the business and system layers. Next, the lowest-level model of the required application is generated with implementation details in it. Finally, this model is transformed into the application code.

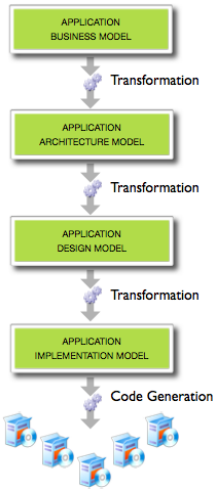

Fig. 1. A Model Transformation Line.

2 The Problem: Extending an MD-SPL

Even though the construction of an MD-SPL requires a high investment, the automatic generation of products makes it a valuable solution to improve software development productivity. An MD-SPLs evolves by offering new features to their family members. For example, it may be necessary to add a new non-functional requirement or crosscutting concern such as security or logging to an existing MD-SPL. While adding a new non-functional requirement we would like to preserve the original MTL. Changing the original MTL may have high-impact and as a result, the process could become very costly. The impact of changes on an MTL is extremely difficult to measure because of the complexity and fragility of transformations. For example, the addition of security to an existing MD-SPL may require changing the metamodels (to add the new security concepts) and the transformations (to transform the new concepts) at several steps.

If, in an existing MTL, we need to add security concepts to the metamodel, the transformations that were defined using this metamodel could become inconsistent with the new version of the metamodel. Transformations need to be updated to fit to the new metamodel. If a change in a transformation rules set is necessary, it could lead to a cascade of changes in consecutive transformation

rule sets. For example, if a change is applied to the first transformation step, the dependencies among the transformations make the probability of impact in the remaining transformation steps very high. Therefore, a simple change in the first set of transformations could require changes in all the transformations in the MTL.

2.1 Challenges

While adding a new non-functional requirement to an existing MD-SPL we would like to preserve the original MTL. In order to do this, we need to modularize the non-functional requirement independently from the main application using Aspect-Oriented Software Development (AOSD) techniques. Moreover, using these techniques we need to tackle the following four challenges:

1. To reduce the impact of changes in the original MTL as much as possible. It is necessary to minimize the cost in adapting the existing MTL for the new crosscutting concern.
2. To offer a high-level mechanism to specify the crosscutting concern independently from the application model. We want to model the new crosscutting concern using high-level concepts and specify it separately from the business model.
3. To provide a reusable mechanism to define relationships between the concern model and the application model at high-level, and to bring the high-level relationships to the low-level. The main objective of having high-level relationships is to reduce the number of manually defined relationships (at a high-level of abstraction fewer concepts are normally defined than at a low-level of abstraction).
4. To offer a transparent and reusable composition mechanism to compose the concern and the application models. To achieve this, it is necessary to identify the most suitable moment in the transformation process to preserve the original MTL. Furthermore, the final composed application model should contain the necessary amount of details required to generate the complete application code with the new concern.

3 Related Approaches

In current research we can find four different approaches that fail to fulfill all these challenges.

High-level Composition In [3], an approach called Model-Driven Security is presented. The goal of this work is to model an application and its security requirements using high-level modeling languages. The entire application together with the access control infrastructures is generated from high-level models, in one transformation. In this approach is impossible to compose a new crosscutting concern with the base application model at the initial transformation step without changing the whole MTL: the original transformations will be useless to manage the security and the application at the same time.

Low-level Composition In order to avoid discarding the MTL, the best moment to compose the new concern and the base application, is after the lowest level model was generated (before the transformation to code). This model contains all the platform-specific implementation details for the application and could be composed with a concern low-level model. This approach is based on traditional Aspect-Oriented Programming (AOP) [4]. The main drawback is the complexity of defining the low-level concern model and its relationships with the low-level application model.

Mixed-level Composition An approach that can be applied to specify a crosscutting concern at a high-level of abstraction and that preserves an existing MTL is used in [5]. This work presents an approach to modularize transactions as aspects and specify them with a high-level transaction language. The high-level transaction aspects are related to a low-level application model using a set of pointcuts. Next, the transaction aspect is transformed and a low-level aspect is generated. Finally, the low-level aspect is woven with a low-level application model. The problem in this approach is the complexity of defining the pointcuts between two different levels of abstraction.

One-step parallel transformation In the work presented in [6], the most suitable approach to our problem is presented. This work presents an approach to define business rules on an application as aspects. The business rules are modeled using a high-level modeling language. The relationships between the business rules and the application are defined using a high-level connection language. This connection language abstracts the different patterns of how the business rules are connected with the application code. Next, both models are transformed to low-level models. The connections are also transformed and aspect code is generated from them. Despite being a good approach, the one step transformations, and the impossibility of defining additional connection between the medium-level models, constitute major problems for its use to tackle the presented problem.

4 Proposed Approach: Multi-step Concern Transformation

The drawbacks of the approaches presented in the previous section show the necessity of using a different approach. Therefore, we propose the Multi-Step Concern Transformation approach in order to fulfill the challenges mentioned in section 2.1. This approach, is based on a parallel transformation of the application model, the concern model, and the relationships between both models. With the proposed approach, the original MTL is preserved and a new transformation line for the crosscutting concern is built into the MD-SPL. In Figure 2 a MTL for security is added and transformed at the same time with the application.

The approach starts from high-level models expressing the application and the concern concepts. In the example, a model for the application is defined

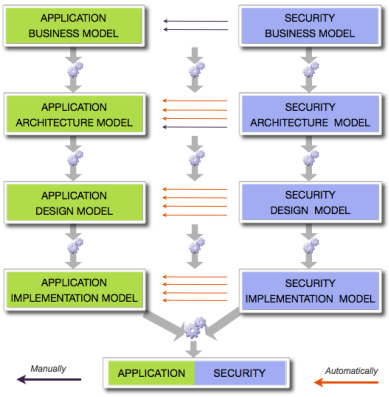

Fig. 2. Multi-step Concern Transformation

using concepts like business entities. On the other hand, the security model is defined with concepts such as resources, roles and permissions. We use a high-level modeling language (metamodel), in order to specify the concern model. After specifying both high-level models, it is required to define manually the relationships between them. For example, these relationships define business entities that are protected resources.

With both models and the relationships defined, several transformation steps are applied to them. At each transformation step, the abstraction level is decreased and new details are added to the application and concern models. In Figure 2, the security model is transformed in the same four steps as the application. At the end a model of the application using Java concepts is generated and the security is expressed with JEE security annotations. In this approach, we postpone the actual composition to the final step of the transformation.

In addition to the model transformation, the relationships are also transformed in every step. In this transformation, details are added to the existing relationships and new ones are created automatically for the new concepts added to the models. This relationship transformation is a complex task because it is necessary to automatically create new relationships between the new elements added to the application and concern models. Moreover, it is possible to define new relationships at each level of abstraction including specific details when the required concepts are already added to the models. Therefore, the support of automatic transformations to manage the relationships between the models reduces the workload in the relationships definition.

Finally, the composition between the application and the concern is executed and a low-level model with the application and the concern concepts is generated. This composition requires managing the transformed models and the accumulated relationships.

5 Progress and Expected Contribution

This work is still in its early stages. The analysis of the possible approaches to tackle the presented problem and the proposed approach were presented in the Early Aspects workshop at AOSD 2008 [2]. The work on progress aims to identify the most adequate mechanism to define relationships between the concern and the application model, such as dialects [3], correspondence models [7], etc. The expected contributions are summarized below:

- A formalization of parallel model transformation.
- To enrich the knowledge of MD-SPL and the association of non-functional requirements to features.
- A methodology to build a crosscutting concern transformation line and add it to an existent MD-SPL

6 Conclusions

This paper has outlined the work on Multi-step Concern Transformation. It has described the problem and context, which is how to add a new crosscutting concern to an existent MD-SPL. It addresses specifically the issue of how to maintain the original metamodels and transformations unchanged and add a new modularized set of transformations for the concern.

References

1. Garcés, K., Parra, C., Arboleda, H., Yie, A., Casallas, R.: Variability management in a model-driven software product line. Revista Avances en Sistemas e Informática **4**(2) (2007) 1–10
2. Yie, A., Casallas, R., Deridder, D., Straeten, R.V.D.: Multi-step concern refinement. In Proceedings of EA workshop in conjunction with AOSD'08. Brussels, Belgium, March 2008 (2008)
3. Basin, D., Doser, J., Lodderstedt, T.: Model driven security: From uml models to access control infrastructures. ACM Transactions on Software Engineering and Methodology (TOSEM) **15**(1) (2006)
4. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J.: Aspect-oriented programming. Proceedings European Conference on Object-Oriented Programming. Jyväskylä, Finland (1997) 220–242
5. Fabry, J., Tanter, É., TD'Hondt: Kala: Kernel aspect language for advanced transactions. Science of Computer Programming (2008) 165–180
6. Cibran, M., D'Hondt, M.: A slice of mde with aop: Transforming high-level business rules to aspects. Proceedings of the 9th International Conference on MoDELS/UML. Genova, Italy (2006) 170–184
7. Bezivin, J., Bouzitouna, S., Fabro, M.D., Gervais, M.P.: A canonical scheme for model composition. Second European Conference, ECMDA-FA 2006, Bilbao, Spain (2006) 346–360