A DSL for Definition of Model Composition Operators[1]

Ivan Kurtev, Marcos Didonet Del Fabro

ATLAS Group, INRIA & LINA, University of Nantes
{ivan.kurtev | marcos.didonet-del-fabro}@univ-nantes.fr

1 Introduction

Aspect Oriented Software Development (AOSD) aims at handling crosscutting concerns in artifacts created during the software development process by improving the modularity constructs and compositional operators in the software languages. The principle of separation of concerns forms the foundation of AOSD. When applied during the modeling phase this principle leads to a number of models of the same system created from different points of view. The models must be properly composed in order to obtain a complete model used for simulation, test derivation, and code generation.

To solve the problem of model composition several approaches have been proposed. Generally, they provide additional compositional operators that do not originally exist in the modeling language being used [2, 4]. There are two problems with these approaches. First, the set of compositional operators is fixed. If the problem at hand cannot be solved with them, the software engineer must resort to an ad-hoc solution. Second, the proposed approaches are based on UML which is regarded as a standard modeling language. However, with the emergence of Model Driven Engineering, the role of Domain Specific Modeling Languages (DSMLs) becomes more and more important. Therefore, we are faced with the need to compose models expressed in various languages. It is hardly to believe that a fixed set of compositional operators coupled with a single language can be used to solve the compositional problems in MDE.

Model composition can be implemented as a model transformation that takes two (or more) input models and generates an output being the composition of them. However, this approach is very general and still ad-hoc. We need an approach independent of any modeling language that allows definition of an open set of reusable compositional operators. Yet, such an approach should provide direct support for common tasks found in model composition such as forming couples of corresponding elements, merging, union, adding, and connecting model elements.

In this position paper we propose a vision for a DSL for definition of model composition operators. The elements of the DSL provide basic operations used to build more complex ones. The DSL is defined as a metamodel that extends the ATLAS Model Weaver [1, 3]. The semantics of the DSL is given by transforming the models expressed in it to executable model transformations.

Section 2 outlines our approach and describes the elements of our DSL. Section 3 gives an example. Section 4 presents conclusions.

2 Primitives for building Model Composition Operators

The approach for definition and usage of compositional operators expressed in the proposed DSL is illustrated in Figure 1.

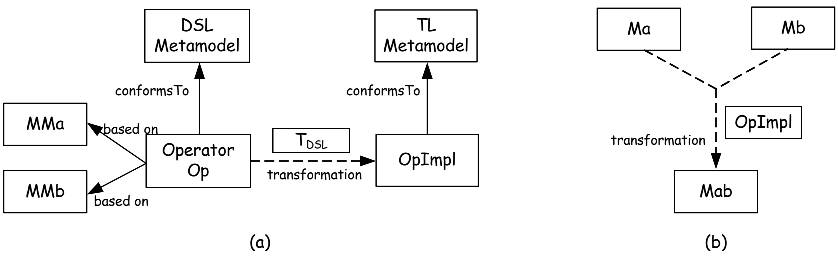

Figure 1. Definition and usage of compositional operators

A compositional operator (*Operator Op* in Fig.1a) is a model that conforms to the DSL metamodel. The operator is based on a number of metamodels (*MMa* and *MMb* in the figure). Defined in that way the operator cannot be directly executed. To obtain an executable implementation the operator is transformed to a model transformation. The transformation that plays the role of compiler is performed by T_{DSL} that may be expressed in any model transformation language. The result *OpImpl* is a model transformation that may be executed over source models. The usage of *OpImpl* is illustrated in Fig.1b where two models *Ma* and *Mb* conforming to the metamodels *MMa* and *MMb* respectively are transformed to a model *Mab*. *Mab* is the composition of the two source models.

[1] Work partially supported by ModelWare IST European project 511731 and a grant from Microsoft Research, Cambridge, UK.

Our DSL for defining compositional operators has a set of primitives defined in the *DSL Metamodel*. The following list enumerates them and sketches their meaning:

- *Select*: selects model elements of a given type based on a given criteria in a given context;
- *Relate*: forms Cartesian product of two sets of model elements eventually selecting tuples satisfying a condition. This operator is very similar to the join operation in relational algebra and comes with the variants for outer joins;
- *Merge*: produces a new model element on the base of two given model elements. By default all the properties of the source elements are copied in the result. The values of properties with the same name are united. This default behavior may be altered;
- *Create*: creates new model elements in the resulting model;
- *Union*: unites sets of model elements;
- *Override*: replaces the value of a property with another value;

3 Example of Package Merge Operator

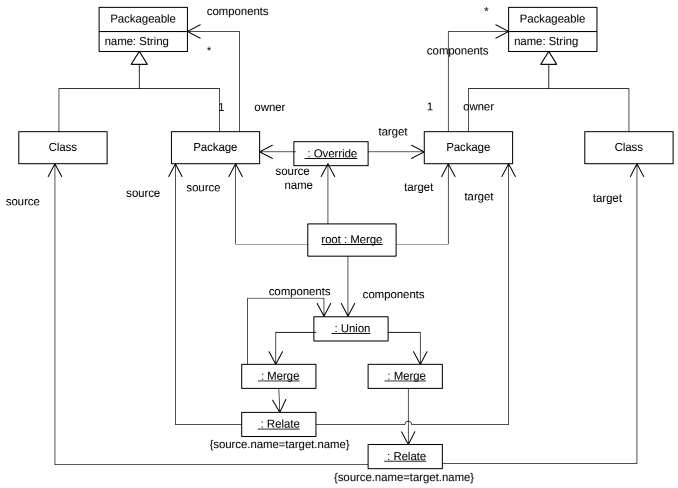

Figure 2. Definition of package merge operator

Figure 2 shows a simple version of package merge operator. It takes two UML packages and merges them. We use a very simplified version of the package construct that may contain classes and other packages. For better visualization the metamodel of *Package* is duplicated. The compositional operator always has a unique root element. In the example it is of type *Merge*. The name of the source package overrides the name of the target package. This is indicated by the *Override* operator owned by the root. The role *name* shows that the override affects the property *name*. The role *components* affects the value of the *Package* property *components.* We apply a union on the results of merging classes and packages owned by the source and target packages. *Relate* operators form Cartesian product based on criteria shown in curly braces.

4 Conclusions

In this position paper we proposed a vision for defining model composition operators based on a DSL. Composition operators are compiled to model transformations and may be reused. The approach is independent from the modeling language used to express the models being composed. To transform this vision to a working solution we have to give a precise semantics to the set of compositional primitives. The approach should be validated by experimenting with non-trivial existing compositional operators and by defining new operators.

References

1. Bézivin, J, Didonet Del Fabro, M, Jouault, F, and Valduriez, P. Combining Concerns with Models. In: First Workshop on Models and Aspects - (MDSD), ECOOP 2005

2. Clarke, S., Walker, R. Generic Aspect-Oriented Design with Theme/UML. In Filman, R., Elrad, T., Clarke, S., and Aksit, M. (eds.) Aspect-Oriented Software Development. Addison-Wesley, 2005

3. Didonet Del Fabro, M, Bézivin, J, Jouault, F, and Valduriez, P. Applying Generic Model Management to Data Mapping. In: Proceedings of Bases de Données Avancées (BDA05), 2005, Saint-Malo, France.

4. Straw, G., Georg, G., Song, E., Ghosh, S., France, R., and Bieman, J. Model Composition Directives, Proceedings of the 7th UML Conference, Lisbon, Portugal, October 10-15, 2004