Transformation Composition Modelling Framework

Jon Oldevik

SINTEF Information and Communication Technology
Forskningsveien 1, 0373 OSLO, NORWAY
jon.oldevik at sintef.no
http://www.sintef.no

Abstract. When applying transformation technologies in an enterprise, there will be a need for supporting compositions of different kinds of transformations to support a development process. An example is a chain of transformations that supports a process of going from requirements to use cases, from use cases to a PIM architecture model, further to a platform specific model and finally implementation code. Some transformation steps may also involve human intervention, e.g. in a refinement of the PSM model, or a detailing of the use case model. This work in progress paper investigates how the atomic transformation viewpoint can be enhanced with support for transformation compositions, to support model driven enterprise process needs. This is done by introducing a modelling framework for composed transformations, based on a hierarchy of transformation types, some of which represent simple atomic transformations, others that represent complex transformations.

1 Introduction

Model transformation is an essential ingredient in model-driven development. In order to support the automation of system development, standards for transformation is emerging in the model-driven development community, driven by the Object Management Group (OMG). The forthcoming transformation standards such as the OMG MOF Query/View/Transformation (QVT)[1] is a good baseline for leveraging the model driven processes of enterprises. When applied in real use, needs will emerge to support the development scenarios in an enterprise, which is often complex and involves a set of integrated modelling, transformation, and validation tasks. An example development process, involving specification of requirements, use cases, architecture models, platform specific models and generation of code, will potentially involve a number of automated or manual transformation tasks, as well as analysis tasks such as model consistency checks.

This paper investigates how to support compositions of transformations that provide the needs that might occur in a model driven development scenario using UML 2 modelling techniques.

2 Transformation Composition Modelling Strategy

The transformation composition modelling strategy aims to support the construction of complex transformations that uses other atomic transformations. Assumedly, for a given enterprise, there will exist a number of defined transformations, which do simple or complicated transformation from a single domain to another.

These transformations might be combined to support parts of, or a complete, development process for the enterprise. The basis for the transformation modelling strategy is a number of transformation types which allows transformations to be combined. A transformation can involve for example a model to model transformation using QVT (or other transformation technology), a model refinement, which is done manually, and model to text transformations.

Doing a transformation in itself might not always be sufficient. A transformation imposes relationships that should be maintained and checked. Changes to models might require transformations to be reapplied. Resulting models might need to be check for consistency and validity. These kinds of analysis activities may also be part of a broader transformation framework.

2.1 Transformation Types

The heart of the transformation framework is the transformation types, which define transformation types with different natures. A *GenericTransformation* represents the common aspects of all types of transformation (Figure 1).

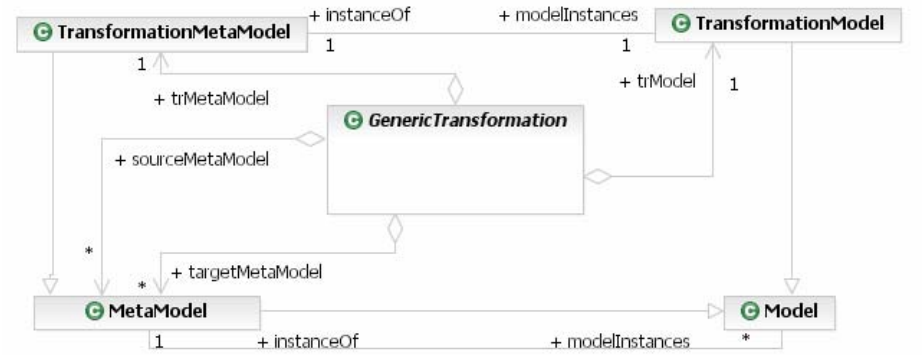

Figure 1 Generic Transformation Structure

A GenericTransformation defines the overall structure and behaviour of all transformations. It is associated with a set of source and target metamodels. It has a transformation metamodel and a transformation model. During runtime, it consumes input (which may be a model) and produces output (which also may be a model).

The input and output provided by a *GenericTransformation* can be of different kind. They may be models, in case of a model to model transformation. Each of them may also be some kind of text (e.g. code), in case of a model to text or text to model transformation.

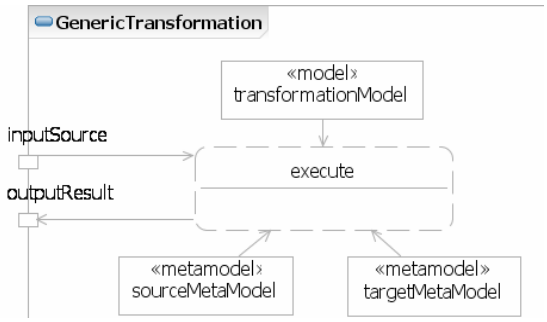

Figure 2 Execution structure of a GenericTransformation

The *GenericTransformation* is specialised into different types of transformations, each of which has a special purpose. The framework is open for additional extensions that provide tailored transformation types (Figure 3).

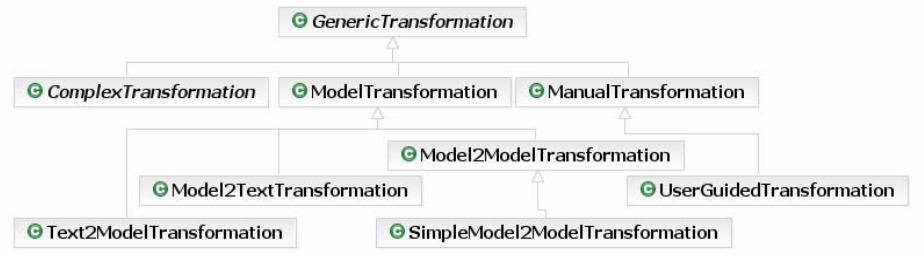

Figure 3 Transformation Types Hierarchy

The transformation hierarchy defines the following specialisations:

- ModelTransformation, which represents automatic transformations involving models either as source or target or both.
- Model2ModelTransformation, which represents a transformation where involving models as both source and target of the transformation. It may include several source or target models.
- SimpleModel2ModelTransformation, which is a special case of Model2ModelTransformation, represents model transformations with only one source and one target model.
- Model2TextTransformation, which represents a transformation from a set of source models to text output.
- Text2ModelTransformation, which represents a transformation from a text to a set of models.
- ManualTransformation, which represents transformations that involves human intervention (i.e. not supported by transformation tools), e.g. manual refinement of a model.
- UserGuidedTransformation, a special kind of manual transformation which is guided by user advice (input), but is executed by a tool.
- ComplexTransformation, which represents more complex transformations (Figure 4)

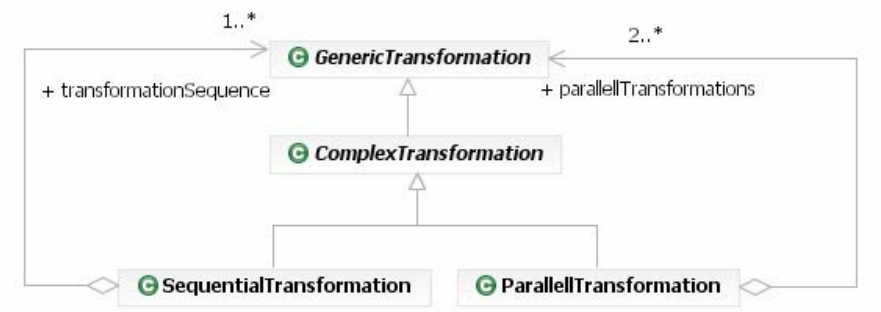

Figure 4 Complex Transformation Types

The *ComplexTransformation* represents transformations that involve several simpler (atomic) transformation tasks. Two types of complex transformations are identified:

- ParallellTransformation, which represents transformations where the referenced transformations can/must be executed in parallel. A parallel transformation references two or more transformations, which are part of the parallel semantics.
- SequentialTransformation, which represents transformations where the referenced transformations must be executed in sequence. A sequential transformation references one or more transformation, which are part of the sequential semantics.

The goal of the *ComplexTransformation* types is to be able to construct useful composite transformations, which basically invokes already existing, well-defined transformations.

2.2 Building Composite Transformations

The composition of transformations requires that there already exists some reusable transformations to compose from. An assumption is taken that a library of existing transformation is readily available. These are described in terms of simple UML 2 activities, which consume input and produces output. The transformations are represented as activity nodes, with input objects (models and metamodels) and output models. Object Flows are used to model the flow of models and metamodels used by the transformation. Activity Parameters denote the consumed and produced input by a transformation.

Figure 5 depicts two transformations; a model to model transformation, going from a Use Case model to a Platform Independent Model (PIM); a manual transformation, refining a PIM model. The transformation themselves are not detailed. They are references, and can be e.g. a QVT specification.

Figure 6 depicts an example of a composition. It is a *SequentialTransformation*, which contains references to the transformations defined in Figure 5, plus an additional PIM2PSM transformation, and a parallel transformation containing two text transformations. Each of the referenced transformations may themselves be more composites, thus allowing for arbitrary complexity of transformations.

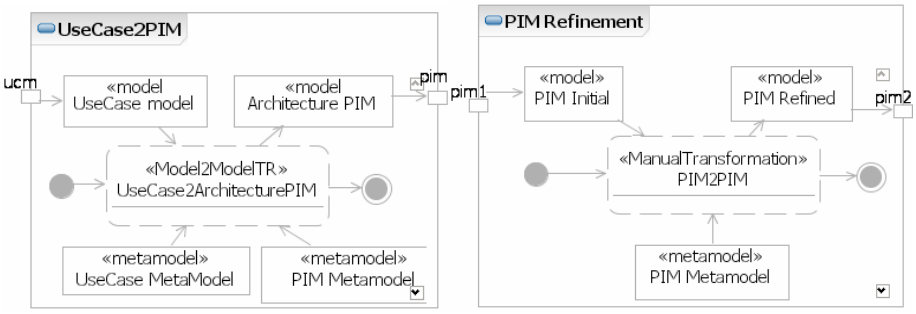

Figure 5 Use Cases to Platform Independent Architecture Model (PIM)

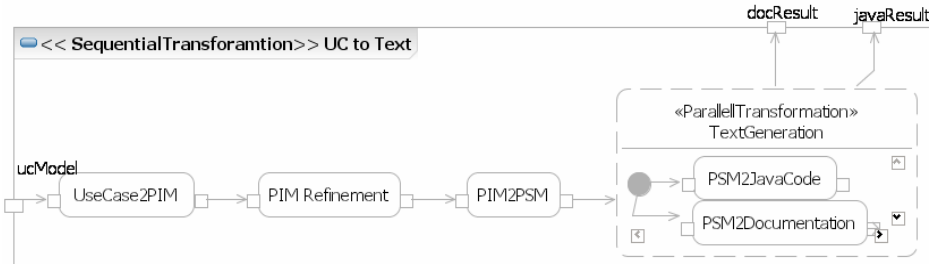

Figure 6 Composite Transformation – Use Case through Text

The process of executing a complex transformation is basically an orchestration task, i.e. making sure that each transformation is executed in its rightful order. In addition, transformation processes may require different kinds of analysis to be done as part of the process. Analysis tasks are also a necessary part of model-driven development chains. Thus, specific tasks handling various pre-defined or tailored analysis of models or text, should have its place in a modelling and transformation framework. This includes aspects such as model conformance, model completeness, traceability, validation, and other analysis tasks relevant for the models or text artefacts consumed or produced. This could be incorporated at a modelling level to support a more holistic model-driven approach.

2.3 Relationship with QVT

The forthcoming QVT language [1] can be used to specify complex transformations in its lexical notation. It also supports composition of transformations through various reuse mechanisms, such as extension or black-box reuse of transformation libraries.

Given a set of basic transformations that stands on their own, it is possible to create a composite transformation, e.g. through the *access* mechanism of QVT.

```
transformation UC2PSM (in ucm : UseCaseMM, out psm: PSM):
access transformation UC2PIM (in ucm : UceCaseMM, out pim : PIM);
access transformation PIM2PSM (in pim : PIM, out psm: PSM);
main () {
   var pimResult := UC2PIM(ucm).transform();
   psm := PIM2PSm(pimResult).transform();
}
```

Combined with control constructs (like conditional branches), structured transformation compositions can be created. The graphical QVT syntax might be used for similar constructions, although the current graphical notation is tuned towards specifying relations between the concepts of a single transformation. A UML2-oriented approach with composite activities (or even composite structures) can leverage specification of higher-order transformations.

3 Related work

In [2], a framework for composition of transformation is proposed, which defines a pattern for composite transformation and a lexical language for defining composites, which handles configuration (execution ordering) of transformation components. This framework does not propose any UML 2 coupling or relate to QVT. In [3], the side transformation pattern is introduced to provide a means of describing combinations of reusable transformations with particular focus of coping with application-specific metamodel incompatibilities. The approach uses the graphical UMLX notation, resembling the graphical QVT notation. Within web service orchestration, choreography and composition [4], a lot relevant of work and technology exist that is pertinent for describing reusable compositions of transformations, such as the Business Process Modelling Notation (BPMN), the Business Process Execution Language (BPEL).

4 Summary and conclusion

This work in progress paper has described ongoing work in the ModelWare project (IST Project 511731)[*], concerning how to handle complex structures of transformations, using a model-based approach for composition of atomic transformations. The continuation of the work will focus on elaborating the modelling framework and support it with tools which are integrated with standards like QVT.

References

1. QVT-Merge Group, Revised submission for MOF 2.0 Query/Views/Transformations RFP version 2.0, OMG document id ad/2005-03-02, http://www.omg.org/cgi-bin/apps/doc?ad/05-03-02.pdf
2. Raphaël Marvie, *A Transformation Composition Framework for Model Driven Engineering*, LIFL 2004n10, November 2004
3. Willink, Harris, 2004, The Side Transformation Pattern, paper at the Software Evolution through Transformations (SETra) 2004
4. Peltz C., *Web Service Orchestration and Choreography*, Web Service Journal Feature, July 2003

[*] MODELWARE is a project co-funded by the European Commission under the "Information Society Technologies" Sixth Framework Programme (2002-2006). Information included in this document reflects only the author's views. The European Community is not liable for any use that may be made of the information contained herein.