On the Specification of Model Transformations through a Platform Independent Approach

Magalhães, A.P.
Exact and Earth Science Department
State University of Bahia
Salvador, Brazil
anapatriciamagalhaes@gmail.com

Andrade, A.; Maciel, R.S.P.
Science Computer Department
Federal University of Bahia
Salvador, Brazil
{aline,ritasuzana}@dcc.ufba.br

Abstract— **Transformations are key artifacts in the MDD (Model Driven Development) approach: a software development project can be defined through a transformation chain converting source models into target models until code, enabling development process automation. Transformations can be complex and demand software processes, languages and techniques to improve their development in order to increase reuse, portability, correctness, and so on. In this context we propose a framework to develop model transformations using MDD. This paper presents a Model Transformation Profile (MTP) defined as the domain specific language of the framework.**

Keywords-Transformation profile, transformation specification, transformation metamodel.

I. Introduction

Model Driven Development (MDD) [7] is a paradigm that makes intensive use of models to represent systems at different level of abstraction (specification, design and code). A key element of the MDD approach is the transformation chain which is responsible for the conversion of source into target models until code generation. Transformations play an important role in MDD because they enable the automation of the model generation process, encapsulating knowledge and strategies used in the development of the software.

Despite the importance of models for the MDD approach, transformations are usually specified in an ad-hoc way using natural language and are implemented directly in code [4]. This practice leads to poor documentation which hampers the evolution of the transformation and makes it difficult to use software engineering good practices such as design patterns and reuse. In order to change this scenario some works have been proposed [2][3][4] to cover specific aspects of transformation development (e.g. transformation design).

In this context, we propose a MDD framework for model transformation development that comprises: (i) a MDD transformation development process, which guides developers through activities to produce transformation software; (ii) a profile, named Model Transformation Profile (MTP), to support the modeling process activities; and (iii) a tool to partially automate the modeling and transformations tasks of the process. In this paper we present the MTP profile whose first ideas were outlined in [6]. The MTP profile presented here has been improved from that incorporating another abstraction level, $\text{MTP}_{\text{LowDesign,}}$ for the specification of transformation behavior. New concepts and attributes have also been added in the other levels and we have developed a validation using experimental software engineering techniques to measure the quality of the profile.

MTP provides concepts to specify model transformations from requirements to design independent of platform. The produced transformation models can be transformed in a specific platform and then in code in different transformation languages (e.g. QTV [9], ATL [1]), increasing productivity and portability. MTP raises the abstraction level of the transformation development from code to model in a platform-independent way, abstracting some implementation details of specific transformation languages.

The rest of this paper is organized as follows: section 2 discusses the current development approaches to model transformation; section 3 briefly introduces our MDD framework; section 4 describes the MTP profile giving some examples; section 5 presents some results from a MTP validation; and finally Section 6 presents our conclusions.

II. Related Works

Our framework uses a visual UML profile as a modeling language, so we focus on comparing our proposal with existing visual approaches. Furthermore, we attempt to analyze the coverage of these works concerning the phases of a transformation development life cycle.

In MOF Query/View/Transformation (QVT) [9] a model transformation can be represented diagrammatically in two ways: using the UML class diagram or using a transformation diagram. The complexity of the QVT metamodels makes the diagram verbose and difficult to understand and the transformation diagram brings new notation with no portability to UML tools.

There are some works that focus on specific aspects of the transformation development. In [3] the authors propose a visual, formal, declarative specification language (graph based) focusing on transformation correctness, but it does not deal with implementation as we do. The work [14] focuses on internal composition of transformations. It generalizes composition mechanisms for rule-based transformation languages in order to provide executable semantic to them. Our proposal, on the other hand, works with external composition. In [15] generic programming is used to define reusable model

(DOI reference number:10.18293/SEKE2015-053)

transformations. We follow another direction through a UML profile to support the development of model transformation models independent of platform such that models are reused in transformations in different languages.

The works [4][2] are more closely related to the one presented in this paper. TransML [4] proposes a family of languages with diagrams for the entire development life cycle providing support for specification, analysis, design and code. However, the proposed diagrams use a UML heavy extension and new notations that make it difficult to integrate with the existing UML tools which are usually adopted. MeTaGen [2] proposes metamodels for transformation design and tools generate code automatically or semi automatically. The main difference between this work and ours is that it focuses on design, not considering the requirement specification level and it uses textual language for transformation specification whereas we use a profile that is a visual language.

In summary, although existing works agree that transformation development requires a software life cycle, they usually focus on an individual phase of development lacking an entire process to transform the transformation model in code. We propose an integrated framework with a visual modeling language specialized from the UML standard that covers transformation development from requirements to code.

III. MDD Transformation Framework

The main goal of MDD Framework is to provide a process to develop model transformations suitable for a transformation domain, covering the entire software development life cycle integrated into a standard modeling language. Fig. 1 shows its main elements: (i) the MDD Transformation Development Process; (ii) the Model Transformation Profile (MTP); and (iii) a tool to (partially) automate the process.

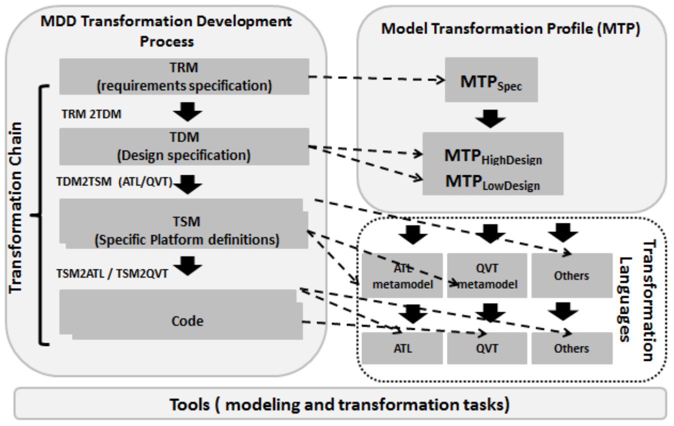
MDD Transformation Development Process
Model Transformation Profile (MTP)
TRM (requirements specification)
MTP_Spec
TRM 2TDM
TDM (Design specification)
MTP_HighDesign
MTP_LowDesign
TDM2TSM (ATL/QVT)
TSM (Specific Platform definitions)
TSM2ATL / TSM2QVT
Code
ATL metamodel
QVT metamodel
Others
ATL
QVT
Others
Transformation Chain
Transformation Languages
Tools (modeling and transformation tasks)

Figure 1. MDD Transformation Framework overview

The MDD Transformation Development Process aims to guide developers step by step on the development of model transformations. The process is specified according to SPEM [8] metamodel and comprises tasks that lead from requirements specification until code. Specification starts modeling the TRM (*Transformation Requirements Model)* which comprises requirements and analysis tasks. From requirements a semi-automatic transformation generates the first release of the TDM (*Transformation Design Model*) which aims to model the design and architecture of the transformation software. Tasks include the definition of *what* might be transformed in what (high design), transformation structure (architecture) and *how* transformation should be performed (low design). This specification is then transformed into TSM (*Transformation Specific Model*) which refers to specific languages to then generate code. We provide generation for TSM in ATL language, due to its wide use in MDD projects to develop transformations, or QVT language, the OMG standard to design model transformations. The MTP Profile is defined to support the modeling tasks of the proposed process. It is detailed in the following sections.

IV. Model Transformation Profile (MTP)

The MTP Profile is a modeling language that extends UML for the model transformation domain. Its main goal is to provide a platform-independent visual language, suitable for a model transformation domain which can be used to develop model transformations at a high abstraction level (TRM and TDM models). The profile covers the definition of model-to-model unidirectional transformation using a visual language.

In order to specify the MTP we define: an abstract syntax, represented by metamodels, with the concepts of the transformation domain; a static semantic, described with a set of OCL constraints which determine the well-formed criteria of the instantiated models; and a set of stereotypes and their UML specialized metaclasses. MTP is divided into three parts, MTP_{Spec}, $MTP_{HighDesign}$ and $MTP_{LowDesign}$.

The main goal of the MTP_{Spec} is to provide definitions for the specification and analysis of transformation requirements. Its abstract syntax is shown in Fig. 2.

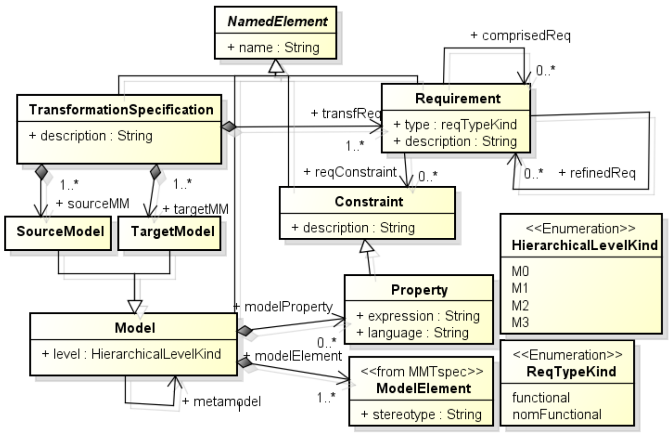

Figure 2. MTP_{Spec} Metamodel

At specification level a *TransformationSpecification* has a *name*, a *description* and is composed of a *Requirement*. *Requirement* may be refined in other requirements (*refinedReq* association) and may also be composed of other requirements (*comprisedReq* association). *Constraint* can be specified for requirements in natural language. A *Requirement* has a *name*, a *description* and a *type* that identifies if it is functional or non-functional. *TransformationSpecification* is also composed of source (*sourceMM*) and target (*targetMM*) metamodels. Models, metamodels and metametamodels are represented by the concept *Model* and have a *level*. This *level* indicates the

OMG model layer in which they are defined (e.g. M3). Properties of specific domains can be specified in *Property*.

The concrete syntax of the *MTP* consists on a package of stereotypes associated to UML metaclasses. For example, the *TransformationSpecification* MTP concept is specialized as an *actor* in UML. Due to lack of space only part of the concrete syntax of MTP$_{spec}$ is shown in Tab.1.

MTP$_{Spec}$ supports the Transformation Process enabling requirements elicitation and analysis in Use Case and Classes diagrams.

TABLE I. PART OF MTP$_{SPEC}$ STEREOTYPE AND METACLASSES

Stereotype	Metaclass
<< Transformation Specification>>	Actor, class
<<Requirement>>	Use case, class
<<Model>>	Class, attribute, package

MTP also comprises a set of OCL constraints with additional well-formed criteria used on model instantiation. Due to lack of space they are not presented here.

MTP$_{Design}$ provides the necessary definitions for the design and architecture specification of the transformation. The profile was organized in two packages, named *MTP$_{HighDesign}$* and *MTP$_{LowDesign}$*.

MTP$_{HighDesign}$ defines *what* will be transformed in what. Its abstract syntax is presented in Fig. 3.

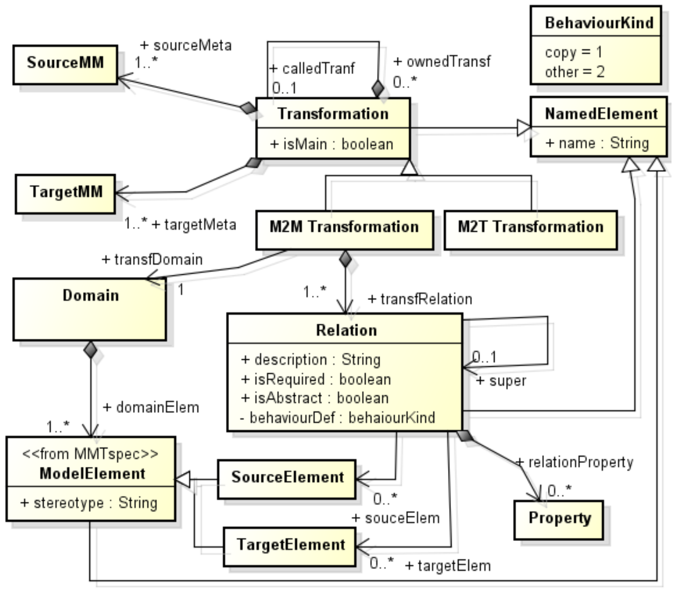

Figure 3. MTPhighdesign metamodel

A *Transformation* may be composed of other transformations, enabling reuse. *Transformation* is specialized in *M2M Transformation*, to represent model-to-model transformations and *M2T Transformation*, to represent model-to-text transformations (not detailed in this work). *M2M Transformation* defines a *Domain* and it is composed of *Relation*. A *Domain* specifies which *Element* of the source/target metamodel will be considered by the transformation. It will be used to verify transformation completeness (section 4A). A *Relation* has a *name*, a *description* to document it and might be concrete or abstract (attribute *isAbstract*) allowing *Relation* inheritance. The attribute *isRequired* indicates if it is automatically processed when transformation is executed or if it is explicitly invoked by another *Relation*. A *Relation* may also have a set of *Property* (e.g. OCL constraints). The main purpose of a *Relation* is the definition of relationships between elements from source to target metamodels (*SourceElement* and *TargetElement*). It is possible to define many kinds of relationships: zero-to-one; zero-to-many; one-to-one; one-to-many; many-to-many; one-to-zero and many-to-zero as shown by the multiplicity of the sourceElem and targetElem association.

MTP$_{HighDesign}$ supports the TDM (Transformation Design Model) specification through the use of classes and component diagrams. Class diagrams are used for the specification of *Relation* between elements from source to target metamodels in order to hierarchically organize the rules of a transformation, providing transformation inheritance. Component diagrams are used to model transformation chains: each transformation in a chain is represented by a component whose interfaces specify the source and target models and metamodels.

The *MTP$_{LowDesign}$* defines *how Relation* converts elements from the source model into elements of the target model. Fig. 4 shows the *MTP$_{LowDesign}$* metamodel. The *Relation* concept (from *MTP$_{HighDesign}$*) is now detailed by the *Rule* concept which is composed of *SourceElementRule and TargetElementRule*. For each *SourceElement* of *Relation* a *SourceElementRule* is modeled for the corresponding *Rule* and a reference (*ref* attribute) must be defined. This reference will be later used in expression definitions. A *SourceElementRule* may be associated to *Condition* (defined in the *exp* attribute) that must be satisfied for the rule to be executed. *TargetElementRule* comprises a set of *Configuration* that defines how the Attributes of the *TargetElementRule* will be initialized when generated. The *Configuration* is specified through the definition of an expression (*exp* attribute) that will be assigned to attributes of the associated *TargetElementRule*. Expressions are defined using a textual language. *MTP*$_{LowDesign}$ supports transformation process through the use of class diagrams.

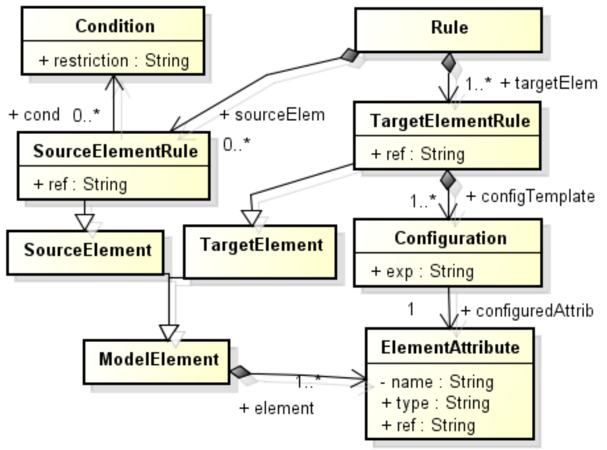

Figure 4. MTP$_{LowDesign}$ metamodel

A. *MTP and Transformation Properties*

There are some properties that assure transformation quality, such as syntactic and semantic correctness and completeness [10][11].

The syntax correctness defines the conformity between models and metamodels and the semantic correctness consists of property preservation from source to target models. We define some OCL constraints in order to guarantee conformance of model transformation models which are instances of MTP. Our framework foresees the specification of semantic properties through the *Property* concept (Fig.2). Therefore it is possible to specify a set of properties in the early stages of the transformation definition and this set can be extended with other properties at the application level.

A transformation is complete if and only if for each element of the source metamodel there is a corresponding element in the target metamodel mapped by the transformation. In order to address completeness, MTP provides the *Domain* concept (Fig.3) which identifies the set of elements of source/target metamodels that are mapped by the transformation. Based on the *Domain* definition and on OCL constraints, completeness can be verified after the instantiation of the model transformation model.

V. MTP VALIDATION

The validation consists in the assessment of the expressiveness of MTP profile constructors. To assist validation we followed the guidelines for software engineering experimentation presented in [13] and use GQM [12] to summarize our goal (Fig.5). The questions underlying the validation are: Q1: Are the MTP constructors sufficient to specify transformations written in ATL/QVT? Q2: Is it necessary to add new constructors in MTP to enable the transformations specification written in ATL/QVT? Q3: Are the selected UML diagrams sufficient to specify transformations?

Analyze the MTP profile constructors
For the purpose of evaluating expressiveness
With respect to coverage of the profile constructors and specification completeness
From the perspective of transformation developers
In the context of existing transformations developed in ATL/QVT languages

Figure 5. Experiment goal according to QGM template

We use several measures as dependent variables such as the amount of used constructors, the need of new constructors, the amount of changes on existing constructors, the level of specification detail and the used UML diagrams.

The validation process lasted five months and was divided into two stages: an *initial test* and the *main validation.* These two stages were performed by our research group in laboratory and consisted of using MTP to specify transformations already developed in ATL / QVT languages.

According to validation, related to questions Q1 and Q2, we concluded that MTP constructors are sufficient to specify transformations without the necessity to add new constructors. Related to question Q3 we observed that, after including component diagram in the *initial test*, the selected UML diagrams were sufficient to specify the transformations. Therefore, we considered that the MTP was stable enough to be used on the framework case study.

VI. CONCLUSIONS AND FUTURE WORKS

The Model Transformation Profile presented in this paper is a modeling language that is part of a framework to develop model transformation using MDD.

MTP represents transformations concepts at different abstraction levels, covering many phases of transformation development such as requirements, analyses and design enabling transformation modeling independent of platform. In this sense it postpones specific platform definitions to later phases of development. As a UML profile MTP takes advantage of the wide use of UML in both industry and academy benefiting from tools already used by the development community. The validation of the profile demonstrated that MTP concepts cover most transformation specification needs and that UML diagrams were suitable for transformation specifications. Therefore, we consider MTP to be stable for use in real projects.

We are currently specifying a MTP behavioral semantics in order to enable simulation of transformation specification.

REFERENCES

[1] ATL Project - http://www.eclipse.org/m2m/atl/

[2] Bollati, V., Vara, J., Jiménez, A., Marcos, E. “Applying MDE to the (semi-)automatic development of model transformations.” Information and Software Technology, pp.699-718, Elsevier, 2013.

[3] Guerra, E.; Lara, J.; Kolovos, D.; Paige, R. “A Visual Specification Language for Model-to-Model Transformations.” IEEE Symposium on Visual Languages and Human-Centric Computing, DOI 10.119/VLHCC.2010.25, 2010.

[4] Guerra, E.; Lara, J.; Kolovos, D.; Paige, R.; Santos O. “TransML: A Family of Languages to Model Model Transformations.” Models, 2010, DOI 10.1007/s10270-011-0211-2, Springer Verlag, . 2010.

[5] Iacob, M., Steen, M., Heerink, L.: “Reusable Model Transformation “ Pattern. In 3M4EC´08, pages 1-10, 2008.

[6] Magalhães, A. P., Andrade, A. ; Maciel, R.S.P.. “MTP: Model Transformation Profile.” In: SBCARS, 2013, Brasilia. p. 109-118 2013.

[7] Mellor,S.; Clark, A.; Futagami, T. “Model Driven Development” IEEE Software,2003

[8] OMG. Software Process Engineering Metamodel Specification, Version 2.0, (formal/08-04-01).2008.

[9] QVT specification - http://www.omg.org/spec/QVT/1.0/PDF/

[10] Lano, K.; Clark, D. “Model Transformation Specification and Verification.” The 18th International Conference on Quality Software, IEEE, 2008.

[11] Mens, T.; Gorp, P.V. “A Taxonomy of Model Transformation.” Elsevier Eletronic Notes in Theiretical Computer Science 152 pp. 125-142 2006.

[12] Solingen, R. Basili, V.;Caldiera,G.; Rombach, H.D. Goal Question Metric (GQM) Approach. John Wiley & Sons. Inc., 2002.

[13] Wohlin, C. Aurum, A. Towards a decision-making structure for selecting a research design in empirical software Engineering. Empir Software Eng DOI 10.1007/s 10664-014-9319-7. Springer, 2014.

[14] Wagelaar, D.; Tisi, M.; Cabot, J.; Jouault, F. Towards a General Composition Semantics for Rule-Based Model Transformation. MODELS, 2011.

[15] Cuadrado, J.; Guerra, E.; Lara, J. Generic Model Transformations: Write Once, Reuse Everywhere. ICMT, 2011.