A Methodological Approach to Developing Model Transformations

Andrés Vignaga

Universidad de Chile, FCFM, Departamento de Ciencias de la Computación
avignaga@dcc.uchile.cl

1 Introduction

In Model Driven Engineering (MDE) model-to-model transformations are a key means for developing software systems. Manual manipulation of models can become unmanageable and expertise in model manipulation would need to be put in practice again and again. This technique has the potential to save a considerable amount of work from developers, as well as to avoid errors associated with manual processes. The research to be carried out in the PhD thesis here described focuses on the definition of a methodology for developing model transformations.

2 Problem Definition

Model transformations are increasingly seen as key assets in software development, as they encode, ultimately formalize and automate, model manipulation expertise of an organization within its own development process, e.g. specific mechanizable refinement steps and the chain of successive refinements needed to obtain a concrete product. Model transformations become complex software products and they need to be developed and managed with sound software engineering principles: they must be analyzed, designed, implemented, tested and maintained [3, 8]. Current research deals with partial aspects of model transformation development, most notably implementation technology, testing, and in a less degree, design. However, results on research towards a systematic development of model transformations have not been published yet [6]. The problem is that transformation development is currently tackled ad-hoc and integrated methodological support for it is still limited. The purpose of this thesis is therefore to investigate how transformation development relate to application development, and to identify specific techniques and methods enabling transformation development and evolution.

Additionally, two particular issues were identified as closely related to the stated problem. First, there is no dominant programming paradigm; [4] identified different approaches, where the underlying philosophy differs substantially from one to the other, thus affecting how model transformations are conceived. Second, categories of model transformations were reported in [9] exhibiting distinctive characteristics; however, implications on model transformation development of such characteristics were not discussed.

3 Related Work

Only a few results on a life-cycle for model transformations were found in the bibliography. In [3], the problem of identifying techniques enabling model transformation development and evolution was introduced. It also presents the basic ideas of a MDA-like development life-cycle, where transformations should be modeled in a technology-independent fashion and then realized within particular environments. Finally, the notion of models of model transformations is further treated in [2], where a possible realization of such idea is presented and its benefits discussed. In turn, [11] presents a simple method for model transformations development. The method is intentionally presented at a high level of abstraction and its scope includes the stages of execution of a typical model transformation only, and it is not a sketch of a complete life-cycle. Such method is used as a basis for discussing what technology supports each of the presented stages of execution. Finally, an associate team [10] is conducting research on various aspects of MDE. The team's schedule for 2007 includes research on model transformation development, however no results have been published yet.

Model transformation analysis is probably the part of the *construction* life-cycle that has gained less attention. A particular use of model transformations motivated that the definition of guidelines for model transformation analysis was identified as a challenge. On the contrary, [7] suggests that requirements for model transformations should be captured informally, and claims that requirement analysis seems to be considerably simpler than for traditional systems.

In [7], a method for model transformation construction is proposed, focusing on design. Such method is incremental and assumes model transformation design as equivalent to transformation rule definition. It is not detailed and supporting tools are still not available. In turn, another work introduces model transformation design patterns, however they all respond to limitations in ATL.

Research on transformation implementation mainly addresses the proposal of model transformation languages, rather than implementation techniques. Implementation approaches such as imperative, declarative, or hybrid were identified in [4]. In turn, [3] proposes to realize technology-independent transformation designs into technology-specific versions, via higher order transformations. This can be regarded as in the context of an implementation stage.

Approaches to model transformation verification and validation include model checking, formal proof and testing. In [8], a framework for an automatic approach of execution of test cases is presented, based on model comparison. Test cases, however, need to be manually specified by the developer and elsewhere a criterion for selecting test data (i.e. models) is proposed. Other works addresses syntactic correctness, and termination and confluence of model transformations. Finally, in [7], a technique for testing a design of a model transformation is proposed.

Only few works deal with model transformation evolution. [5] presents a strategy for the incremental maintenance of rule-based transformations to address the problem introduced by incremental updates on models during their life-cycle. However, other scenarios of evolution, such as a refinement in the model manipulation expertise, is not addressed.

4 Goal and Research Hypotheses

The main goal of the thesis is to define a methodology specifically aimed at developing and evolving model transformations. Next are stated the working hypotheses which will be addressed.

H1: A method applying MDE techniques can be defined for developing and evolving model transformations.

H2: Applying such method to model transformation development enhances the quality of transformations and the productivity of the development team.

5 Proposed Solution

We propose to solve the stated problem by defining a methodology for developing and evolving model transformations. The focus will be set on design and implementation activities, however the scope shall include the entire life-cycle.

A development process is built on best practices collected throughout the experience of the community. For model transformations, a collection of best practices is still to be completed. To that end, general Software Engineering best practices may serve, at least, as an inspiration. This claim demonstrated to be particularly valid, for example, in model transformation testing. However, adapting existing application development methodologies to the model transformation domain would result unnecessary restrictive. We consider more appropriate to come up with a solution that freely combines established knowledge of traditional development with research in the model transformation area, from an MDE-minded point of view.

The solution will be a full-fledged process expressed as a SPEM model. We propose a *lifecycle* based on an iterative and incremental model, and structured in *phases*; at least one for construction and one for evolution. The scope of the proposed *activities* includes requirements, analysis, design, implementation, testing and management. Activities will be associated to *process roles* and input and output *work products*, organized into *disciplines*, and refined into *steps*. Whenever possible, the proposal shall also provide *guidance* on process elements, especially for activities, steps and work products. Activities and steps will be described in detail, and the procedure for generating output work products from input work products will be made explicit. Work products, in turn, will be precisely described, especially those which will be specific to model transformation development. This enables automatic work product manipulation.

6 Methodology and Validation Strategy

The proposed methodology is intended to apply to the development of practical model transformations. For defining such a development process we need to analyze concrete experiences in the context of concrete applications of development processes [4]. Only a few practical model transformations were reported [12].

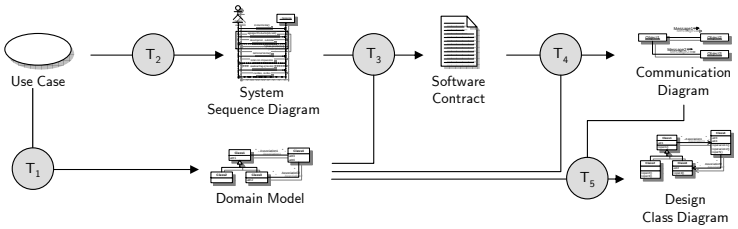

Fig. 1. Partial definition of RUP as a partial order of model transformations.

We therefore propose to use a concrete development process as a source. The Rational Unified Process (RUP) was chosen for a number of reasons. First, it is a modern and widely used development process. Second, "modeling" is one of RUP's best practices, and although it is not defined in terms of transformations, models are core concepts. Third, it is general and broad in scope; we expect to identify from it a large number of assorted model transformations. Figure 1 shows a part of RUP, rendered as a partial order of model transformations.

The set of model transformations identified can then be classified according to criteria based on those introduced in [4] and [9], and classes of equivalence can be defined. Developing an appropriate representative of each class will provide the basis for defining our methodology. Inspiration will also come from related work and from established application development processes, for example RUP itself. We also believe that our methodology would benefit from concepts present in MDE-based processes.

Experiments will be then conducted by applying the methodology to a fresh set of transformations of different classes of equivalence. Some of those can possibly be from outside the scope of RUP. The results will be used for adjusting the methodology as required. To this end, a set of metrics for evaluating the quality of model transformations, such as size, complexity, scalability and maintainability, need to be defined. Validation is finally conducted revisiting some selected classes of equivalence. Using the defined metrics, we compare results of the initial ad-hoc development against those from applying the final version of the methodology.

Concrete activities are: *(Act1)* Investigate characterizations of model transformations and define classification criteria; *(Act2)* Identify a set of practical transformations; *(Act3)* Classify the transformation set; *(Act4)* Develop and analyze the selected transformations; *(Act5)* Investigate existing processes; *(Act6)* Build the methodology; *(Act7)* Define metrics; *(Act8)* Refine and validate the proposal; and *(Act9)* Evaluate the results.

7 Advanced Work and Current State

In what follows, we refer to the activities presented above and discuss their relative progress and status. Investigation in *Act1* was already performed as

part of the survey of related work. *Act2* is partially performed. Figure 1 shows a representation of a part of the existing results. *Act4* is partially performed. Work in [12] and [13] reports the results of development carried out so far, corresponding to transformations $\mathsf{T_5}$ and $\mathsf{T_4}$ in Fig. 1, respectively. *Act5* is close to completion, and results were used for identifying practical transformations. *Act6* was started. With results already obtained, basic decisions about the structure of the methodology have been made.

8 Expected Results

The expected contributions of this thesis are: *(1)* A generic methodology for developing and evolving model transformations; *(2)* Definition of RUP as a partial order of model transformations; *(3)* Definition of criteria for classifying model transformations in multiple dimensions; *(4)* Classification of the transformations identified for RUP into classes of equivalence; *(5)* Implementation of a set of transformations identified for RUP applying the proposed methodology; *(6)* Evaluation of the implemented transformations; and *(7)* A set of metrics for measuring quality aspects of model transformations.

References

1. *Model Driven Engineering Languages and Systems, MoDELS 2006*, volume 4199 of *LNCS*. Springer, 2006.
2. J. Bézivin, F. Büttner, M. Gogolla, F. Jouault, I. Kurtev, and A. Lindow. Model Transformations? Transformation Models! In *MoDELS* [1], pages 440–453.
3. J. Bézivin, N. Farcet, J.M. Jézéquel, B. Langlois, and D. Pollet. Reflective Model Driven Engineering. In *UML*, volume 2863 of *LNCS*, pages 175–189, 2003.
4. K. Czarnecki and S. Helsen. Classification of Model Transformation Approaches. In *OOPSLA 2003 Workshop on Generative Techniques in the context of Model Driven Architecture*, October 2003.
5. D. Hearnden, M. Lawley, and K. Raymond. Incremental Model Transformation for the Evolution of Model-Driven Systems. In *MoDELS* [1], pages 321–335.
6. J. M. Küster. Definition and Validation of Model Transformations. *Software and Systems Modeling*, 5(3):233–259, 2006.
7. J. M. Küster, K. Ryndina, and R. Hauser. A Systematic Approach to Designing Model Transformations. Report RZ 3621, IBM, Zurich, July 2005.
8. Y. Lin, J. Zhang, and J. Gray. A Testing Framework for Model Transformations. In *Model-driven Software Development*, pages 219–236, Chapter 10. Springer, 2005.
9. T. Mens and P. Van Gorp. A Taxonomy of Model Transformation. *Electronic Notes in Theoretical Computer Science*, 152:125–142, 2006.
10. MATT Associate Team. INRIA and Colorado State University. Internet: `http://www.irisa.fr/triskell/matt/`, 2006.
11. L. Tratt. Model Transformations and Tool Integration. *Software and System Modeling*, 4(2):112–122, 2005.
12. A. Vignaga and C. Bastarrica. Transforming System Operations' Interactions into a Design Class Diagram. In *SAC*, pages 993–997. ACM, 2007.
13. A. Vignaga, D. Perovich, and C. Bastarrica. Extracting a Design out of Software Contracts using Model Transformations. In consideration for publication, 2007.