Model Transformation by Demonstration

Yu Sun
Department of Computer and Information Sciences
University of Alabama at Birmingham
Birmingham, AL 35294
yusun@cis.uab.edu

Abstract
A common approach toward model transformation is to write transformation rules in specialized languages. However, their usage may present challenges to those who are unfamiliar with a specific model transformation language or a particular metamodel definition. The research described in this paper makes a contribution toward simplifying the creation of model transformations by recording and analyzing the operational behavior exhibited by an end-user.

Categories and Subject Descriptors I.2.2 [**Artificial Intelligence**]: Automatic Programming; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms Algorithms, Design, Languages.

Keywords Model transformation, demonstration.

1. Background and Motivation
Model transformation has emerged as a core part of Model-Driven Engineering (MDE). Examples of model transformation include code generation from models, model synchronization and mapping, model evolution, and reverse engineering. Several approaches have been developed to perform model transformations, such as: direct model manipulation, intermediate representation, and transformation language support [1].

Direct model manipulation accesses the internal structure of a model instance using an API provided by a host modeling tool, and encodes the transformation procedures in a general-purpose programming language (GPL). This approach is not feasible for end-users who do not have programming experience, because GPLs lack the high-level abstractions that are needed by end-users to specify transformations. In addition, the power of a transformation is often restricted by the supported API within the modeling tool.

Many modeling tools support importing and exporting model instances in the form of XMI. It is possible to use existing XML tools (e.g., XSLT) to perform model transformations outside of a modeling tool using XMI as an intermediate representation. Although XSLT can be used to transform models, it is tightly coupled to XML, requiring experience to define the transformations using concepts at a lower level of abstraction.

A more common and popular approach toward implementing model transformations is to specify the transformation rules by using a model transformation language. Although most of these languages are powerful, they still present several challenges to users, particularly to domain experts and non-programmers. Even though declarative expressions are supported in most model transformation languages, they may not be at the proper level of abstraction for an end-user, and may result in a steep learning curve and high training cost. Furthermore, the transformation rules are usually defined at the metamodel level, which requires a clear and deep understanding about the abstract syntax and semantic interrelationships between the source and target models. In some cases, domain concepts may be hidden in the metamodel and difficult to unveil [2]. These implicit concepts make writing transformation rules challenging. Thus, the difficulty of specifying metamodel-level rules and the associated learning curve may prevent domain experts from contributing to model transformation tasks from which they have much domain experience.

The research described in this paper contributes a new approach to simplify the realization of model transformations, enabling general users (e.g., domain experts and non-programmers) to specify model transformations without knowledge of a specific model transformation language or metamodel definition.

2. Related Work
Model Transformation By Example (MTBE) [3] is an innovative approach to address the challenges inherent from using model transformation languages. Instead of writing transformation rules manually, MTBE enables users to define a prototypical set of interrelated mappings between the source and target model instances. From those mappings, the metamodel-level transformation rules can be inferred and generated semi-automatically. Varró proposed a practical and efficient way to realize MTBE by using inductive logic programming [4], [5]. The basic idea is to represent the initial mappings in the form of logic clauses and then infer the transformation rules using a logic programming engine. Similarly, Strommer and Wimmer implemented an Eclipse prototype to enable generation of transformation rules from the semantic mappings between domain models [2], [6], [7]. Instead of using a logic programming engine, their inference and reasoning process was based on pattern matching.

However, the current state of MTBE research still has several limitations. The semi-automatic generation often leads to an iterative manual refinement of the generated rules; therefore, the model transformation designers are not isolated completely from knowing the transformation languages and the metamodel definitions. In addition, the inference of transformation rules depends on the given sets of mapping examples. In order to obtain a complete and precise inference result, one or more representative examples must be found for users to setup the prototypical mappings, which is not always an easy task in practice. Furthermore, current MTBE approaches focus on mapping the corresponding domain concepts between two different metamodels without handling complex attribute transformations. Finally, the related work mentioned here primarily has been applied to exogenous model transformation, but they are not as beneficial for inferring the refinements that are typical of endogenous model transformations where the source and target models are from the same metamodel.

OOPSLA 2009, October 25-29, 2009, Orlando, FL, USA.
ACM 978-1-60558-768-4/09/10.

3. Goals and Objectives

The Model Transformation By Demonstration (MTBD) research described in this paper further simplifies the model transformation process initiated by MTBE. The contribution of MTBD is a technique that will enable all model users (i.e., not only model experts and programmers, but also domain experts and non-programmers) to specify the desired model transformations, without knowing any model transformation language or metamodel definition. The realization of MTBD has the potential to provide fully automatic generation of transformation rules without manual refinement of a transformation specification. MTBD will also be applicable to both exogenous and endogenous model transformations, enabling complex attribute computations. In addition, MTBD can be applied to any model instance without being restricted by the availability of appropriate source and target models.

4. Proposed Methodology

The core idea is a new technique that records user interactions within a modeling tool and infers a representative model transformation specification. Instead of inferring the rules from a prototypical set of mappings (as done in MTBE), users are asked to demonstrate how the model transformation should be done by directly editing (e.g., add, delete, connect, update) the model instance to simulate the model transformation process step-by-step. The user transforms a source model to the target model during the demonstration process. A recording and inference engine will capture all user operations and infer a user's intention in a model transformation task. A transformation pattern will be generated from the inference, specifying the precondition of the transformation (i.e., *where* the transformation should be done) and the sequence of actions needed to realize the transformation (i.e., *how* the transformation should be done). This pattern serves as an intermediate transformation representation, which can be used to generate different model transformation rules, code, data and other necessary transformation artifacts. The final generated rules and code can be reused in any model instance at any time.

5. Experimental Evaluation

The evaluation of MTBD will be based on three criteria – completeness, correctness and simplicity. Regarding the first two criteria, for each kind of model transformation (i.e., exogenous and endogenous model transformations), some existing transformations written in a specific model transformation language will be selected. For instance, the ATL transformation zoo [8] provides a list of model transformation scenarios that have been implemented by ATL (e.g., Class to Relational, UML to OWL). MTBD will be used to automatically generate the transformation rules. Given the same set of source models, we can compare the target models produced by the two approaches. The similarity between the two sets of target models reflects the completeness and correctness of our approach. The simplicity of MTBD will be evaluated by observing the time and process for applying MTBD, as well as the scale of the transformation rules to realize the same task. For example, the size and complexity of an ATL transformation will be compared to the relative effort (in terms of mouse clicks and time) to specify the same transformation by demonstration.

6. Current Results

The current focus of this work is the implementation of endogenous model transformation by demonstration. Our work is implemented in an Eclipse-based domain-specific modeling tool called GEMS (Generic Eclipse Modeling System) [9]. An Eclipse plug-in has been developed, which partially realizes the MTBD idea in endogenous model transformations. More specifically, the current status of the MTBD prototype includes: (1) a recording engine to completely capture all user operations and related context; (2) an algorithm to optimize the recorded operations, eliminating meaningless operations (e.g., an add operation followed by a delete operation are both meaningless if they operate on the same object); (3) the inference and generation of a transformation pattern from the recorded operations that describe the weakest precondition and the transformation actions; (4) an algorithm to automatically match a transformation precondition in any model instance, and execute the necessary transformation actions; (5) support to infer transformations with attribute operations; (6) a correctness checking and undo mechanism to guarantee the correctness of the transformation process; (7) fully automatic generation of a transformation pattern, without iterative manual refinement.

We have applied our approach successfully to complete several model refactoring tasks in sample domains, demonstrating transformation correctness and simplicity improvement. More detailed description of the examples and representative video demonstrations are provided at the project's web site, which is available at: http://www.cis.uab.edu/softcom/mtbd.

Acknowledgment

This work is supported in part by an NSF CAREER award (CCF-0643725).

References

[1] Sendall, S., Kozaczynski, W.: Model transformation - The heart and soul of model-driven software development. IEEE Software, Special Issue on Model Driven Software Development, vol. 20, no. 5, pp. 42-45, Sep./Oct. 2003.

[2] Wimmer, M., Strommer, M., Kargl, H., Kramler, G.: Towards model transformation generation by-example. In Proceedings of the 40th Hawaii International Conference on Systems Science, Big Island, HI, January 2007, pp. 285.

[3] Varró, D.: Model transformation by example. In Proceedings of Model Driven Engineering Languages and Systems, Genova, Italy, October 2006, pp. 410–424.

[4] Balogh, Z., Varró, D.: Model transformation by example using inductive logic programming. Software and Systems Modeling, vol. 8, no. 3, July 2009, pp. 347-364

[5] Varró, D., Balogh, Z.: Automating model transformation by example using inductive logic programming. In Proceedings of the 2007 ACM Symposium on Applied Computing, Seoul, Korea, March 2007, pp. 978–984.

[6] Strommer, M., Wimmer, M.: A framework for model transformation by-example: Concepts and tool support. In Proceedings of the 46th International Conference on Technology of Object-Oriented Languages and Systems, Zurich, Switzerland, July 2008, pp. 372–391.

[7] Strommer, M., Murzek, M., Wimmer, M.: Applying model transformation by-example on business process modeling languages. In Proceedings of Third International Workshop on Foundations and Practices of UML, Auckland, New Zealand, November 2007, pp. 116–125.

[8] ATL Transformation Zoo. http://www.eclipse.org/m2m/atl/atlTransformations/

[9] Generic Eclipse Modeling System (GEMS). http://www.eclipse.org/gmt/gems/