# MT-Scribe: A Tool for Recording and Inferring Model Transformations

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

Jules White
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37203
jules@dre.vanderbilt.edu

## Abstract

The traditional model transformation approach is to write transformation programs in a specialized language. Although such languages provide powerful capabilities to automate model refinements, they still present challenges to those who are unfamiliar with specialized model transformation languages or metamodels. MT-Scribe is a tool to simplify the creation of model transformations by recording and analyzing user-specified model transformations so that they can be modularized and applied to other models. This approach does not require users to have knowledge of any model transformation language or familiarity with a metamodel definition. Through this demonstration, the audience will observe an innovative approach to implement model transformations and understand how to apply this approach to simplify endogenous model transformations tasks to support model evolution.

***Categories and Subject Descriptors***D.2.2-2.6 [**Software Engineering**]: Design Tools and Techniques; Programming Environments; I.6.5 [**Simulation and Modeling**]: Model Development

***General Terms*** Algorithms, Design, Languages.

***Keywords*** Model transformation, Demonstration, MT-Scribe.

## 1. Introduction

Model transformation plays an essential role in many applications of model engineering using domain-specific languages. The traditional and common approach to realize model transformations is to write transformation rules and automate transformation processes by using an executable model transformation language. Although most existing model transformation languages are powerful enough to implement large-scale and complex model transformation tasks for different purposes, they present some challenges to users, particularly those who are unfamiliar with a specific transformation language. First, since these languages may not be at the proper level of abstraction for an end-user, a steep learning curve and high training cost are inevitable, especially when the users have little programming experience. Moreover, the transformation rules are usually defined at the metamodel level, requiring 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 (e.g., some concepts are hidden in attributes, association ends or enumerations, rather than being represented as first-class entities), which makes writing transformation rules challenging. Thus, the difficulty of specifying metamodel-level transformation rules and the associated learning curve may prevent some domain experts from building model transformations for which they have extensive domain experience.

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

To address the challenges inherent in using model transformation languages, Model Transformation By Example (MTBE) has been proposed [1]. Rather than writing transformation rules manually, MTBE enables users to setup interrelated mappings between the source and target model instances, and the metamodel-level transformation rules are automatically generated. However, the current MTBE implementations [1][2] have limitations that prevent them from being adopted widely. First, the generation of rules is semi-automatic and requires user refinement, so users cannot be fully isolated from the knowledge of transformation languages and metamodel definitions. Also, current approaches focus on directly mapping the corresponding domain concepts between two different metamodels without handling complex attribute transformations (e.g., an attribute in the source model is transformed to another attribute in the target model by some arithmetic or string operations).

To overcome such problems and to further simplify the realization of model transformation, we have been investigating the idea of Model Transformation By Demonstration (MTBD). Tool support, such as presented in this demonstration, can assist general model users (e.g., domain experts and non-programmers) in realizing model transformations without knowing a specific model transformation language or the metamodel definition.

## 2. MTBD Solution

The MTBD idea derives from MTBE. Instead of inferring the rules from a set of interrelated mappings between the source and target models, users are asked to demonstrate how the model transformation should be done by directly editing (e.g., add, delete, update) the source model to simulate the transformation process step-by-step. A recording engine captures all of the user's operations during the demonstration. After the source model is changed into the desirable target model, the system obtains all the necessary operations to realize the transformation task. Then, the inference engine infers the user's intention and generates a transformation pattern from the recorded operations. This generated pattern can be reused and executed by the engine in any model instance to carry out the model transformation.

As an implementation of the MTBD idea, MT-Scribe is an Eclipse plug-in for GEMS (Generic Eclipse Modeling System) [3]. The current version focuses on endogenous model transformations (i.e., both the source and target models conform to the same metamodel). It consists of five main steps, as shown in Figure 1.

***Step 1 - User demonstration and operations recording***. The demonstration is given by directly editing a model instance. At the same time, an event listener monitors all the operations occurring and related context information.

***Step 2 - Optimize recorded operations.*** The sequence of operations recorded directs how a transformation should be performed. However, not all operations are meaningful (e.g., a user first adds a new element and then deletes it in another operation, the result being that both operations cancel each other in the transformation process and therefore are meaningless). An algorithm has been designed to eliminate meaningless operations automatically.

***Step 3 - Infer the transformation pattern.*** A transformation pattern is inferred by analyzing the recorded operations. The generated pattern describes the precondition of a transformation (i.e., where the transformation should be performed) and the actions of a transformation (i.e., how the transformation should be realized).

***Step 4 - Execute transformation patterns.*** The inferred transformation patterns can be reused and executed in any model instances. A backtracking algorithm has been implemented to automatically match a precondition in a selected model instance. After a matching location is found, the transformation actions will be replayed to carry out the transformation.

***Step 5 - Correctness checking.*** Precondition matching guarantees that operations can be executed with the required operands. However, it does not ensure that executing them will conform to the metamodel definition. Each applied operation is logged and model instance correctness checking is performed after every operation execution. If a certain operation violates the metamodel definition, all executed operations are undone and the whole transformation replay is cancelled.

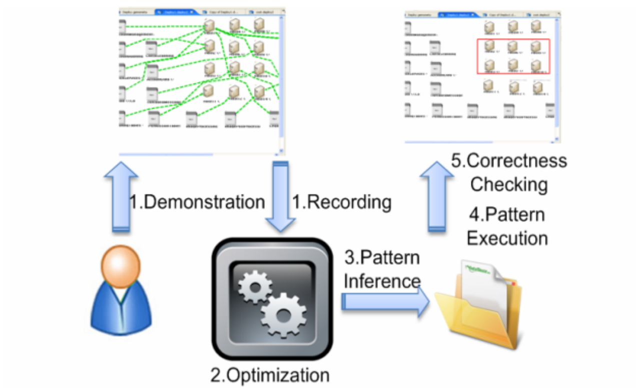

**Figure 1.**Overview of MT-Scribe

Throughout the process, users are only involved in Step 1. All other steps are finished automatically without any manual refinement. Since the user is provided with a high-level domain-specific GUI to specify the transformation, no model transformation languages are used and the generated transformation patterns are invisible to users. Users are completely isolated from the underlying model transformation specification language and the metamodel definition. In addition, complex attribute operations can be inferred, making our approach more powerful compared with the MTBE approaches. This demonstration will illustrate several practical endogenous model transformations in different domains, showing improvement in the efficiency and simplicity of specifying model transformations. Video demonstrations of this process are available at http://www.cis.uab.edu/softcom/mtbd

## 3. Demonstration Overview

### The Presenters

**Yu Sun** is a Ph.D. candidate in the Department of Computer and Information Sciences (CIS) at UAB and member of the SoftCom Laboratory. His research interests include domain-specific modeling, domain-specific languages and model transformation techniques. He is the implementer of the MT-Scribe plug-in. **Dr. Jeff Gray** is an Associate Professor in the CIS Department at UAB, where he co-directs research in the SoftCom Laboratory. His research interests include model-driven engineering, aspect orientation, code clones, and generative programming. Dr. Gray is the designer of the MTBD idea. **Dr. Jules White** is a Research Assistant Professor at Vanderbilt University. His research focuses on applying model-driven engineering and constraint-based optimization techniques to the deployment and configuration of complex software systems. Dr. White is the project leader for GEMS.

### What Will the Audience See

The demonstration will be structured to contain the following parts through both a PowerPoint presentation and a live example of several case studies demonstrating use of MT-Scribe.

***Part 1 - Motivation for MT-Scribe.*** The background information about model transformation and model transformation languages will be given, followed by an introduction to the problems associated with using model transformation languages. We will also briefly introduce MTBE and its limitations to highlight the motivation of MT-Scribe. A transformation sample problem in a simple domain will be used as a motivating example.

***Part 2 - Overview of MT-Scribe.*** We will give an overview of MT-Scribe, including the basic MTBD idea and its main steps. Then, we will demonstrate how to use MT-Scribe to solve the motivating transformation example. The demonstration will explain the main implementation details including the GEMS modeling platform, the architecture of MT-Scribe, and algorithms used.

***Part 3 - More Demos of MT-Scribe.*** The presentation will proceed by demonstrating several endogenous model transformations in two different domains. We will provide MT-Scribe solutions for several common transformation tasks in a modeling language for describing textual games. Another example domain represents UML class diagrams. Some typical UML refactoring tasks such as *Extract Subclass*, *Push Down Method*, *Hide Delegate* will be demonstrated to show how MT-Scribe can simplify these refactoring tasks.

## Acknowledgement
This work is supported by NSF CAREER award CCF-0643725.

## References

[1] Balogh, Z., Varró, D.: Model transformation by example using inductive logic programming. Software and Systems Modeling, DOI 10.1007/s10270-008-0092-1, Springer Berlin / Heidelberg.

[2] 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.

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