Supporting Model Evolution Through Demonstration-based Model Transformation

http://www.cis.uab.edu/softcom/mtbd

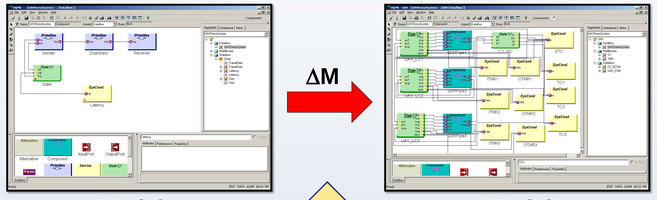
Software Composition and Modeling Laboratory
Department of Computer and Information Sciences
University of Alabama at Birmingham

Yu Sun (yusun@cis.uab.edu)
Advisor: Dr. Jeff Gray

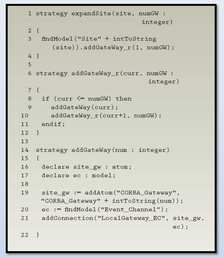
This material is based upon work supported by the National Science Foundation under Grant No. CAREER-0643725

Background

Model Evolution



The traditional approach to automate model evolution is to use a specialized model transformation language (e.g., QVT, ATL, C-SAW)

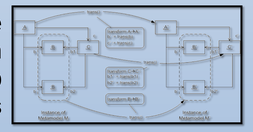


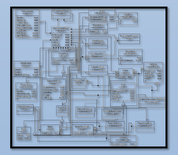
Model Transformation Language

Problems

Using model transformation languages presents challenges:

- Model transformation languages are not at the proper level of abstraction for an end-user; therefore, a steep learning curve and high training cost is inevitable.
- Transformation rules are defined at the metamodel level, requiring a deep and clear understanding about the complex abstract syntax and semantics.

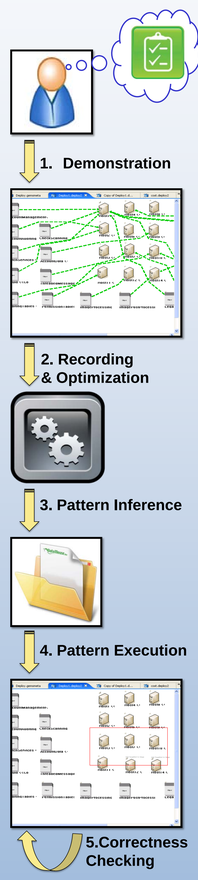




Research Goal

Design an innovative approach to simplify the creation of model transformations, so that general modeling users can easily implement model transformations to support model evolution without the knowledge of a model transformation language or metamodel definitions. The approach to model transformation by demonstration (MTBD) allows an end-user to define the essential transformations in their domain of expertise using notations and abstractions that are familiar to them.

MTBD Solution



Model Transformation By Demonstration (MTBD) simplifies the creation of model transformations by recording and analyzing the operational behavior exhibited by end-users

1. **User demonstrates the transformation process.** The demonstration is given by directly editing a model instance (e.g., add a new model element, modify a model element) to simulate a transformation task on a concrete example.

2. **Record and optimize user operations.** An engine has been developed to monitor and capture all the operations occurring in the model editor, including related context information. The recorded operations will also be optimized in order to eliminate all meaningless operations.

3. **Infer the transformation pattern.** Based on the operations recorded, a transformation pattern is inferred, which 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).

4. **Pattern execution.** By selecting a pattern from the repository, the MTBD engine automatically traverses the model instance to search all locations that match the selected pattern. Once a matching location is found, the recorded operations will be replayed to carry out the transformation.

5. **Correctness checking.** Model instance correctness checking is performed after every operation execution to guarantee that the replayed operation does not violate metamodel definitions.

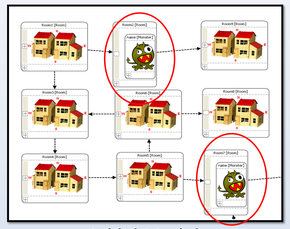
Results and Contribution

MT-Scribe is an Eclipse plug-in to GEMS (Generic Eclipse Modeling System), which implements the MTBD idea. The contributions include:

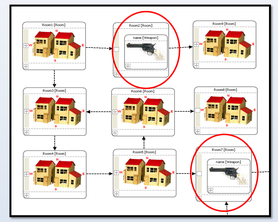
1. A fully automatic inference and generation process to support model evolution (e.g., model refactoring, aspect weaving, and scalability).
2. Users are completely isolated from knowing any model transformation languages or metamodel definitions.
3. Complex arithmetic and string attribute operations are supported, which is a major limitation of other related efforts.

Case Study

A model refactoring example in MazeGame domain:
Replace the *monster* in a room with a *weapon*, and set the *powerValue* attribute of the new *weapon* to half of the monster.

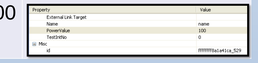


Initial Model



Target Model after Evolution

Find a Room that contains a monster, and perform the demonstration:

Step	Operation	Result
1	Select Monster1 in Room2	
2	Delete Monster1	
3	Add a new Weapon	
4	Set the *powerValue* of the Weapon to 100 (Monster1.powerValue) / 2	

After the demonstration, a transformation pattern can be generated, which can be reused to transform all the rooms that contain monsters.

The same process can be applied in UML refactoring as well, showing improvement of the simplicity of realizing model refactoring tasks to support model evolution.

More examples and MT-Scribe videos are available at the project site:
http://www.cis.uab.edu/softcom/mtbd

Conclusion & Future Work

The primary contribution of this research is an implementation of a simplified model transformation approach, enabling general users to easily realize model evolution tasks.

The future work will focus on applying the MTBD idea on model transformations between two different domains in order to support other model engineering applications, such as model mapping, model interoperability and synchronization.