Transformations to Automate Model Change Evolution

Yuehua Lin (liny@cis.uab.edu)  Jeff Gray (gray@cis.uab.edu)

1. Background

- Domain-Specific Modeling (DSM) addresses the complexity of software development by raising the specification of software to visual models at a higher-level of abstraction.
- MetaModel describes the entities and their relationships in an application domain.
- Models are instances of a metamodel to visually represent software using idioms familiar to domain experts.
- Model Transformation is a process that converts source models to the target models that are the final output.
- Generic Modeling Environment (GME) is a modeling tool that allows one to define a domain-specific visual modeling language.
- ECL transformation specification contains the details of how the transformation engine should transform the source models to target models.
- C-SAW: The Transformation Engine is implemented as a GME plug-in component, with a parser and an interpreter for ECL.
- C-SAW takes source models and the ECL transformation specifications as input, and generates target models as output by weaving changes into source models.

2. Motivation

- MODEL COMPLEXITY IN SIZE & STRUCTURE: Distinguished from traditional class-level UML models, many Domain-Specific models are instance models that usually have repetitive and nested structures, and may contain large quantities of components.
- EXPLORATION OF DESIGN ALTERNATIVES: A powerful justification of the use of Domain-Specific models concerns the flexibility of system analysis that requires an ability to change models rapidly and cheaply. However, manually visiting such models can be labor intensive, time consuming and prone to errors.

Research Goal: Allowing the accidental complexity of modeling large-scale applications by providing a model transformation approach to automate model change evolution.

- Improve the productivity and reduce the potential errors in model evolution.

Areas of Contribution:
- Explore new roles for model transformation in addressing model change evolution issues such as model scalability.
- Provide an easy-to-use model transformation language to specify and execute model change evolutions; a core model transformation engine has been developed to automate such tasks.
- Provide support for testing model transformation specifications. Model differencing techniques are applied to model transformation testing for comparing the actual output and the expected output.

3. Research Overview

The Embedded Constraint Language (ECL) is an imperative transformation language supporting model-to-model transformation.

- It provides operations for model navigation, selection and transformation.
- It has two kinds of modular units:
  - Aspect specifies a crosscutting concern across a model hierarchy (e.g., multiple locations in a model).
  - Strategy specifies elements of computation (e.g., transformation behaviors) which will be bound to specific model elements defined by an aspect.

C-SAW: The Transformation Engine is implemented as a GME plug-in component, with a parser and an interpreter for ECL.

- C-SAW takes source models and the ECL transformation specifications as input, and generates target models as output by weaving changes into source models.

4. Model Transformation Engine: C-SAW

- Modeling Environment (GME)
- ECL transformation specification to scale up a model

5. Case Study: Transformation to Address Model Scalability

- It is often desirable to evaluate different design alternatives as they relate to scalability issues of the modeled system.
- Model scale-up is automated by a transformation process that expands the number of elements from the base model and makes the correct connections among the generated modeling elements.

6. Model Transformation Testing

- C-SAW testing engine performs all the tests for testing a specification, which has three components:
  - Executor executes the to-be-tested specification on the input model to generate the output model.
  - Comparator compares the output model (i.e., the actual result) to the expected model (i.e., the expected result) and collects the results of comparison.
  - Test analyzer visualizes the model differences to assist in comprehending the testing results.

7. Evaluation and Contribution

- Model Scalability: This example shows the capability of C-SAW to scale up a system configuration model by increasing its copies and associated menu from 1 to 9 and building the necessary connections.

Evaluation:
- C-SAW has helped to address model scalability issues for different applications such as embedded avionics, computational physics and performance analysis. Example languages that were used in the evaluation include:
  - The System Integration Modeling Language (SIML) was used to specify configuration of large-scale data processing systems used by satellites.
  - The Stochastic Reward Net Modeling Language (SRNML) assists in modeling system performance of middleware.
  - Other modeling artifacts from the Escher institute.

- Experimental results have shown using C-SAW to perform model change evolution is significantly faster and more accurate than manual processes.
- Future evaluation will be conducted on determining the effectiveness of the testing engine in detecting errors in the transformed results.

Contributions:
- Automating model transformations for evolving models rapidly and correctly.
- Applying software engineering processes such as testing to model transformations.
- Developing algorithms for comparing models in the context of Domain-Specific Modeling.

Acknowledgement:
This project was previously funded by the DARPA Program Competition for Embedded Systems (PCES) program, and currently supported by the National Science Foundation under CNS-0509342.