Barriers to Systematic Model Transformation Testing

Benoit Baudry, Sudipto Ghosh, Franck Fleurey, Robert France, Yves Le Traon, Jean-Marie Mottu

Soroosh Nalchigar
Soroosh@cs.toronto.edu

October 2012
Programming?

Modelling & generating!
Model Transformations

- Model transformation
  - Automatic
  - One or multiple input models
  - One or multiple output models

- Transformation definition
  - Set of transformation rules

- Transformation rule
  - Describe how one or more construct in the source language can be transformed

Kleppe et al. (2003), Mens & Gorp (2006)
Model Transformations

- Vertical versus horizontal
  - Abstraction level

- Endogenous versus exogenous
  - The language in which the source and target models are expressed.

Mens & Gorp (2006)
So why?

Different intentions of doing model transformations:
- Refinement
- Abstraction
- Synthesis (e.g., code generation)
- Reverse engineering
- Refactoring
- Synchronization
- ...
Example of model refactoring

Before:

LeagueOwner
+email:Address

Advertiser
+email:Address

Player
+email:Address

What kind of transformation is this?

After:

User
+email:Address

LeagueOwner

Advertiser

Player
Example of code generation

public class User {
    private String email;
    public String getEmail() {
        return email;
    }
    public void setEmail(String value) {
        email = value;
    }
    public void notify(String msg) {
        // ....
    }
}

public class LeagueOwner extends User {
    private int maxNumLeagues;
    public int getMaxNumLeagues() {
        return maxNumLeagues;
    }
    public void setMaxNumLeagues(int value) {
        maxNumLeagues = value;
    }
}

What kind of transformation is this?
What is this paper about?

Why it is difficult to, systematically, test model transformations?

What are possible ways to overcome these barriers?
A small example!
A small example!

Before
Hierarchical state machine

After
Flattened state machine
How to test model transformations?

- **Step 1:** Generate test data
  - Generate input models (test models)

- **Question:** What other input models would you suggest?
How to test model transformations?

- **Step 2:** Define test adequacy criteria
  - It is not possible to test a transformation with all possible input models
  - E.g. instantiate all the meta-classes

- **Step 3:** Construct an oracle
  - Determines if the result of a test case is correct
  - Could be partial and check expected properties of the output model
Where we are?

Why it is difficult to, systematically, test model transformations?

What are possible ways to overcome these barriers?
Complex input and output data

- Complexity of data manipulated by transformations:
  - Large input and output models
  - Multiple views
  - Large meta-models
  - OCL constraints

- This complexity affects generation of test models:
  - Manual: error-prone
  - Automatic: complex constraint solving problem
Complex input and output data

- Lack of historical data on typical errors:
  - Difficult to determine effectiveness of test adequacy criteria
  - Difficult to determine the fault models they can target

- Complexity of output data complicates oracle problem:
  - To build the expected results
  - Even if the output model is available?
Possible Solutions

- Use of SAT Solvers
- Constructive approach
- Automatic test model generation
- Test the output model directly
- Partial oracles
- Work on requirements definition
- Definition of Oracle
- Partial oracles
- Work on requirements definition
Possible Solutions

- Use of SAT Solvers
  - It has the limitations of current automatic constraint solvers
  - Use of heuristics and meta-heuristics?
- Constructive approach
  - Brottier et al. (2006)
- Test the output model directly
- Partial Oracles
  - Checks only specific properties of the output instead of checking the result completely
- Develop techniques for the precise definition of requirements for model transformations
  - Extract logical formulae from natural languages?
Model Management Environments

- MDE is in the research stage, support is needed for:
  - Building
  - Editing
  - Visualizing, and
  - Analyzing models.

- Manual definition of test models is tedious and error-prone.

- Construction of models involve either:
  - Writing a program: error-prone
  - Using model editors generated from a meta-model:
    - Lack of language-specific icons
    - Lack of dialog boxes for setting attribute values
    - Lack of assistance for checking the completeness
Model Management Environments

- Regression testing:
  - Need for sophisticated model comparison tools

- Visualizing output models is difficult
  - Lack of adequate support for layout of diagrams
Possible Solutions

- Ongoing works could be adapted or integrated for testing model transformations.

- Model differencing:
  - EMFCompare Tool: detects matches and differences between two models based on similarity in their type, name, and values.

- Versioning of models:
  - Used to define oracles that compares models and detect conflicts
  - Detection of test models that are still relevant and those need to evolve.
Heterogeneity of Transformation Languages and Techniques

- Large number of model transformation languages and techniques
- Strong impact on the definition of effective white-box test adequacy criteria
- It is impossible to know if any one of the many techniques will fit all the needs for model transformations.
- We cannot choose one language as a reference and develop test criteria that are based on language elements.
Black-box techniques:

- Fleurey et al. (2007) proposed a set of rules and a framework to assess the quality of test models.
- Assumed that all meta-models are modeled in EMOF.
- Evaluate the quality based on coverage requirements:
  - Class coverage
  - Attribute coverage
  - Association coverage
- Benefit: any transformation language.
- Should consider intent of transformation.
Possible Solutions

- White-box techniques:
  - More detailed and effective than black box, capture mechanics of transformation.
  - Must cover the individual transformation steps. The tester should generate tests models that exercise each steps.
  - Küster et al. (2006) defined a template language to generate test models based on the structure of the transformation rules.
Discussion

- 37 citations from 2010 …

- Does the intent of transformation affects difficulty of tests?

- What about MMTF?

- A suggestion for the course … My be a basic paper with 468 citations?


