BARRIERS TO SYSTEMATIC MODEL TRANSFORMATION TESTING

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

Presented By: Lobna AbuSerrieh

Contents

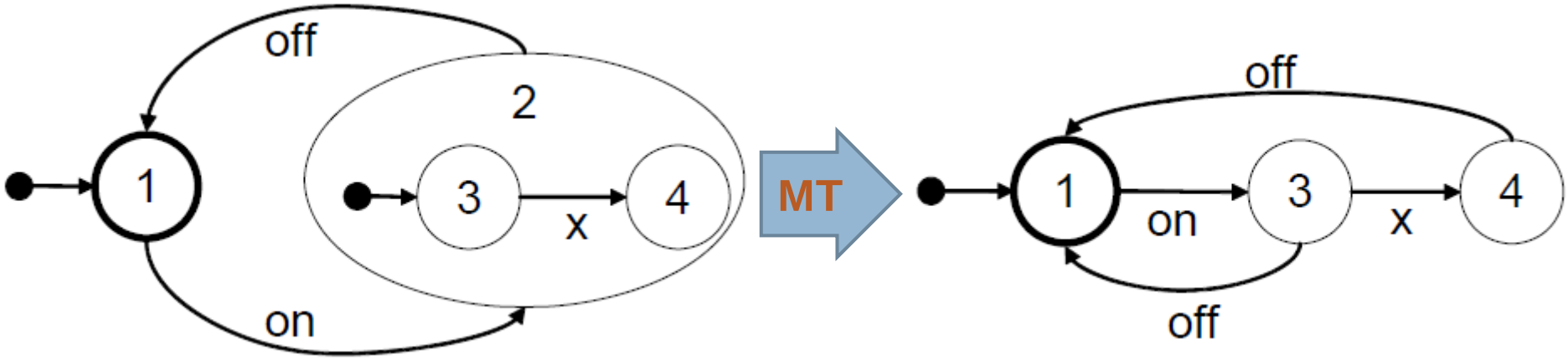
- Model Driven Engineering
- Model Transformation Testing
- Example
- Characteristics / barriers of Model Transformation testing
- Approaches to overcome these barriers

Introduction

- **Model Driven Engineering (MDE) :**
  - Models constitutes the basic units of the development.
  - Automated Model transformation plays critical role in MDE.
  - Airbus uses automatic code generation from SCADE models for embedded controllers in Airbus A380.
  - Objecteering: UML and MDA CASE tool which supports MDE.

Model Transformation/ Example

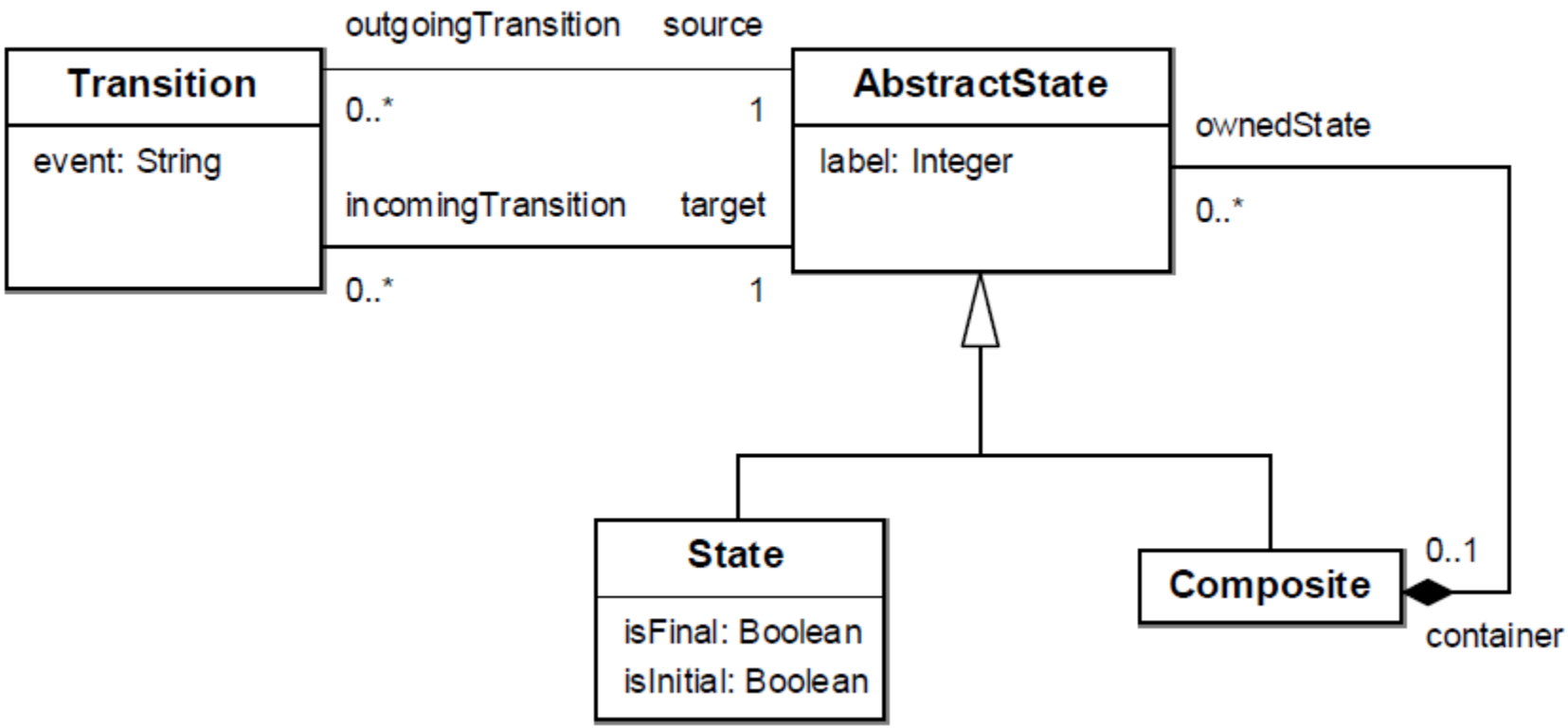
- Flattening a state machine



a. Hierarchical state machine

b. Flattened state machine

Model Transformation/ Metamodel



A Hierarchical State Machine Metamodel

Model Transformation/ OCL

- It is usually necessary to define constraints more precisely; And it must be added to the metamodel.
- OCL is commonly used to define additional constraints.

Context: Composite

Inv:

self.ownedState →select(AbstractState as | as.oclIsTypeOf(State))

→ select (AbstractState s | s.oclAsType(State).isInitial) →size()=1

Model Transformation Testing

- The correctness of transformation is essential to the success of MDE.
- A fault in transformation can introduce a fault in the resulted transformation model.
- Since model transformations are meant to be reused, faults present in them may result in many faulty models.

Model Transformation Testing/ Activities

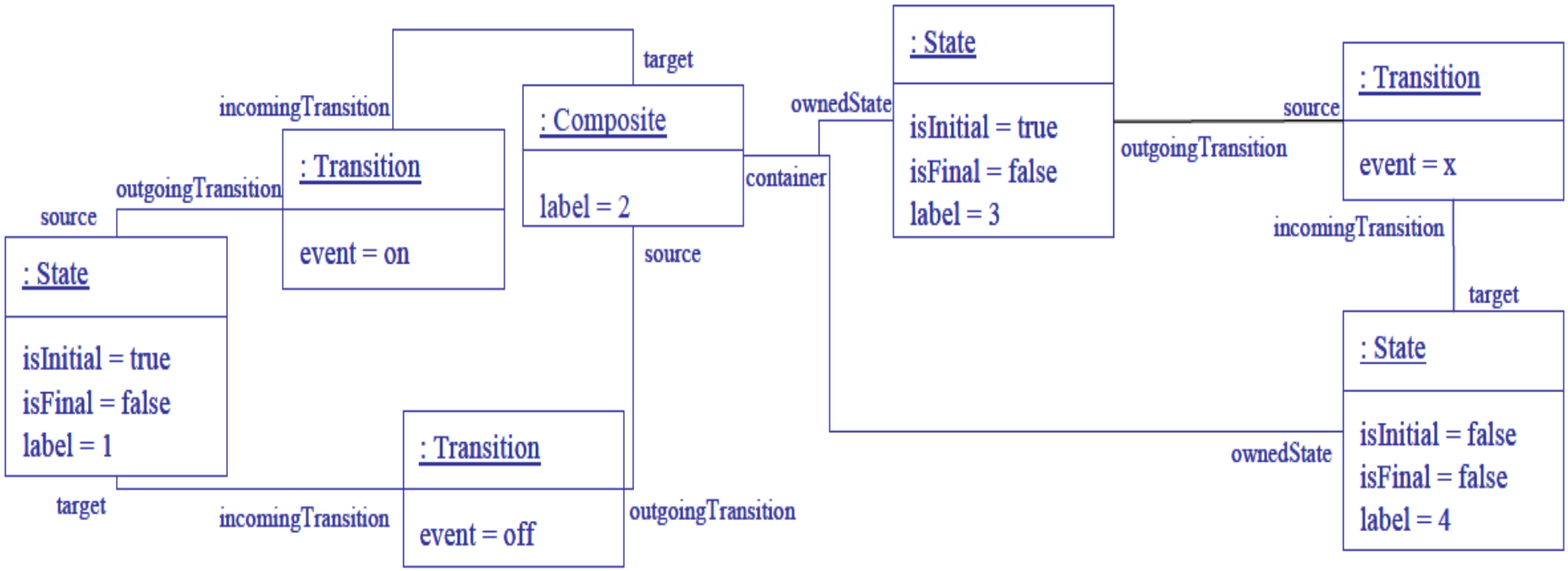
- Testing :

  Prepare the Input – Run – evaluate the Output

- Testing Model Transformation:

1. Generate test data
2. Define test adequacy criteria
3. Construct an Oracle

MTT/ Generate test data



Test Model for the Flattening Transformation

Characteristics of Model transformation

- Model transformation has some unique characteristics which make testing it challenging:

1. Transformation Input and output Complexity.
2. Model management tools.
3. Various Transformation languages.

Characteristics of Model transformation

- Model transformation has some unique characteristics which make testing it challenging:

1. Transformation Input and output Complexity.
2. Model management tools.
3. Heterogeneity of Transformation languages.

Complexity of Input and Output data

- Models are often large.
- The metamodels can themselves be large & complex.
- Additional constraints using OCL increases the metamodel complexity.
- OCL is a rich language with which it is possible to define complex constraints relating a large number of elements in the metamodel.

Complexity of Input and Output data/ Input Data

- This complexity affects the generation of test models.
  - **Manual test data generation** is error-prone.
  - **Automatic test data generation** is a complex constraint solving problem;
- It is possible to define a large number of test adequacy criteria;
- However, lack of historical data makes it difficult to determine the effectiveness of these criteria and the fault models they can target.

Complexity of Input and Output data/ Output Data

- Output Complexity complicates the oracle problem.
  1. When the expected output model is available, the oracle needs to compare two models. Then the oracle problem complexity is NP-complete.
  2. If the oracle is specified by listing expected properties of the output model, then building this oracle is complicated by the complexity of the output metamodel that describes the output model.

Characteristics of Model transformation

- Model transformation has some unique characteristics which make testing it challenging:

1. Transformation Input and output Complexity.
2. Model management tools.
3. Heterogeneity of Transformation languages.

H/Model Management Environment

- The construction of models involves either:

  1. **writing a program** that builds the metamodel instances. Or

  2. **using model editors** to manually build the instances, e.g. EMF.

- Visualizing output models is difficult because graphical editors often do not provide adequate support for layout of diagrams that are produced by a transformation.

- A confusing layout complicates manual analysis and the comparison of two graphical representations. Specially with regression test.

Characteristics of Model transformation

- Model transformation has some unique characteristics which make testing it challenging:

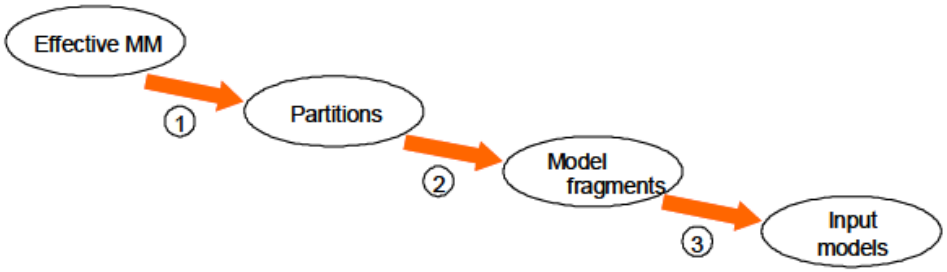
1. Transformation Input and output Complexity.
2. Model management tools.
3. Heterogeneity of Transformation languages.

Heterogeneity of Transformation Languages

- A Large number of model transformation languages and techniques exist.
- Transformations can be implemented with general purpose programming languages; or languages dedicated to model transformations (e.g. QVT). In addition to tool-specific transformation languages, e.g. Objecteering, MetaEdit+.
- Testing techniques need to take this diversity into account.

Promising Approaches/ Input Complexity

- A constructive approach where models are built first and the constraints are checked afterwards
  - Generating objects and assemble them according to specific criteria in order to build complete models.



- **Limitation:** Large number of of generated models do not satisfy complete set of constraints
- Use SAT solvers to deal with a larger amount of constraints and generate instances that satisfy the constraints.

Promising Approaches/ Output Complexity

Dealing with the Oracle Complexity:

1. When generating a model; test the output model directly.
2. Using partial oracle that checks only specific properties of the output.
3. Using patterns to express pre- and post-conditions for the transformation.
4. Using "Design by Contract" when building a model transformation.

Promising Approaches/ Model Management Environments

- Model differencing: compares the model produced after execution of a test case with an expected model.
  - **EMFCompare** tool is available in the Eclipse framework.
- Versioning of models can benefit testing.
  - **CVS Model** is an open source initiative that proposes a tool for versioning of models.

Promising Approaches/ Heterogeneity of Transformation Languages

- Dealing with this issue can be tackled by:

1. Specific criteria and associated test generation techniques for each particular language.
2. Black box techniques that ignore the actual language used for the transformation.
3. A white-box approach generates test models based on the structure of the rules used to implement the transformation.

Conclusion

- Some of the major challenges are identified.
- Solutions to some of the testing problems exist, but need more improvement and work.
- A benchmark of realistic models and model transformations for validation and comparison purposes is needed.

Questions and Discussion