MODEL TRANSFORMATION TESTING, THE STATE OF THE ART

Gehan Selim, James Cordy, Juergen Dingel,

Presented by: Lobna AbuSerrieh
INTRODUCTION

Model Driven Development

Model

Transformations

Code

Programming

Modeling → Generating
Transformation Correctness

Formal Methods: Heavyweight

Testing:

- executes a transformation on input models then validates the actual output matches the expected output.
- Automatable test activities
- Lightweight, Low computational complexity
PHASES OF MODEL TRANSFORMATION TESTING

1. Test Case Generation
2. Test Suite Assessment
3. Building the Oracle
4. Execute and evaluate
Phase 1: Test Case Generation

- Define test adequacy criteria, then Build test cases that achieves its coverage. And it can be done by using:
  - **Black-Box testing**: based on transformation specification.
  - **Gray-box testing**: based on the accessible parts of transformation implementation.
  - **White-Box testing**: based on transformation implementation.
Black-Box Test Case Generation
Meta Model Coverage

- Adequacy criteria for Class diagrams
  - Association end multiplicity criterion
  - Generalization criterion
  - Class attribute criterion
Black-Box Test Case Generation

Meta Model Coverage

- Adequacy criteria for Interaction diagrams
  - Each message on a link
  - All message path
  - Collection coverage
  - Condition coverage
  - Full predicate coverage
  - Transition coverage
Adequacy criteria for statecharts

- Full predicate coverage
- All content-dependency relationships
- Transition coverage
- Transition-pair coverage
- Complete sequence coverage
- All configurations transition coverage
Black-Box Test Case Generation

Contract Coverage

Achieving input contracts of Model transformation

- Constructing metamodel of only those elements are actually used in pre/post conditions of transformation

- Combine contract-based and metamodel based. And footprints (number of times test model covers each criterion).
**White-Box Test case generation**

- Most of the Studies are done without case studies and no detailed results.
- Transforming rules to a source meta model template.
- Assessing ATL rules by profiling:
  1. Compilation resulted XML file to extract the rules.
  2. Transformation to be executed. And using the resulted log file to assess the coverage (rule, instruction, decision).
- Grammar testing, Each rule to be triggered in every possible context.
**Phase 2: Test Suite Assessment**

- Achieved Coverage to assess the test suite quality.

- Mutation analysis, evaluate the sensitivity of the test case to faults in transformation.

- Injecting faults by applying mutation operators and generate mutants.
  - Different results: Killed mutant.
  - No faults: the mutant is alive
Phase 3: Building the Oracle Function

Compared: The actual output with expected one.

- if the expected output is available, then Compare.
- If it is not available, validates the resulted output with the predefined output properties or contracts.
Phase 3: Building the Oracle Function

Comparison

If the expected output is available, then compare:

- **Test Case Constructor**
  (Input Model, Expected Output, Transformation Strategy)
- **Test Engine**
  Execute, Compare
- **Test Analyzer**
  Visualize using colors and shapes

A framework uses Model comparison
PHASE 3: BUILDING THE ORACLE FUNCTION

CONTRACTS

If the expected output is not available, validates the result with the predefined output properties or contracts.

- Tracts, set of OCL constraints and a tract test suite.

- Improving Transformation contracts:
  1. Vigilance: dynamically detect errors
  2. Diagnosability: effort to locate a fault
Phase 3: Building the Oracle Function Contracts

- Vigilance can be improved by analyzing a test suite and repeatedly using mutation analysis, until achieving an acceptable mutation score.

- Other proposed an improved vigilance and diagnosability by using mathematical modeling.
Questions

- Gray-Box Testing, is it feasible to depend on partial implementation while considering other parts as black box testing?
- Class diagrams, statecharts, and sequence diagrams are the common used while testing transformation, what about other types of diagrams?
- Is Model comparison as oracle function clear enough?
- Since 2012 when this paper was written, and many related studies were without case studies or reliable results, any new updates were added to testing MDT?