Model transformation testing: oracle issue

Presented by Philip Mitchell

Jean-Marie Mottu[1] Benoit Baudry[1] Yves Le Traon[2]

IRISA
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
{jmottu, bbaudry}@irisa.fr

TELECOM Bretagne
2 rue de la Chtaigneraie CS 17607
35576 Cesson Svign, France
yves.letraon@telecom-bretagne.eu

March 26, 2012

Outline

Model transformations

- Used for model-driven development (MDD)
- Automate critical operations:
  - refinement
  - code generation
  - refactoring

Verifying model transformations

- Need to verify correctness of transformation
- Two main concerns when choosing test cases:

test data	a set of inputs to the model transformation that will exercise as many code paths as possible
oracle function	a function that analyzes the validity of models produced by the model transformation

- We will focus on defining the oracle function

Defining an oracle function

- Model transformations produce models that conform to a metamodel
- Models resulting from a transformation can be very complex
- Need to definitively determine whether the resulting model from a test transformation is correct
- This generally requires at least some feedback from the tester

Model transformation

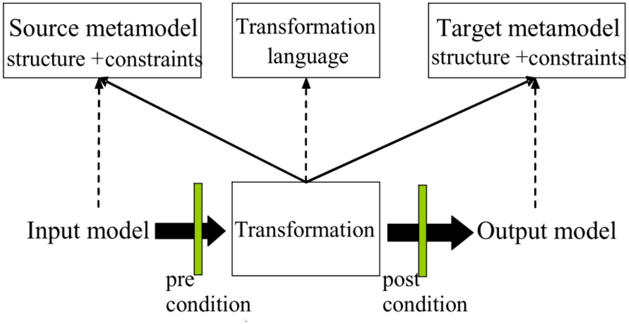

Figure 1: General framework for model transformation T

Model transformation

- Given an input model conforming to the source metamodel with constraints
- Produce an output model conforming to the target metamodel with expected properties
- Illustrative example used throughout this presentation:
    - transform a class model to an RDBMS model as proposed in the MTIP workshop
    - requires complex operations on the input model, recursivity, navigations with transitive closure, and several passes
    - output model conforms to the RDBMS metamodel

Input model for illustrative example

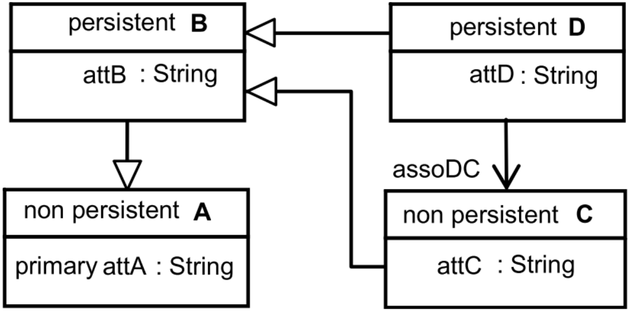

Figure 2: Input model *Mt*1

Details of the illustrative example

- Additional constraints apply to the input class model, for example:
  - Each class should have at least one primary attribute
- Additional constraints apply to the RDBMS metamodel, for example:
  - An RDBMS model cannot contain two tables with the same name
- The transformation rule for T is as follows:

  Ru The persistent classes, and only these ones, are transformed into tables with the same names, except if they inherit directly or not from another persistent class.

Output model for illustrative example

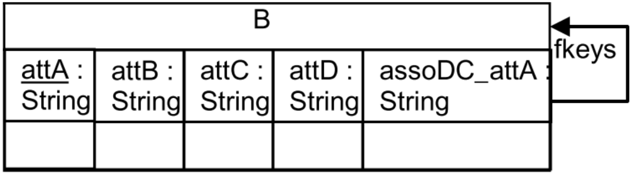

Figure 3: Output model from $T(Mt1)$

Output RDBMS metamodel for illustrative example

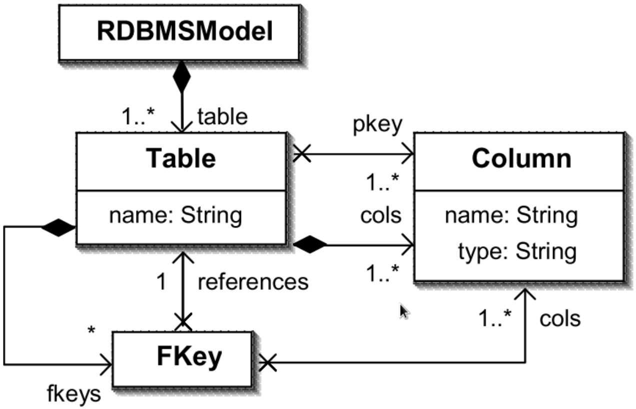

Figure 4: Output RDBMS metamodel

Testing the transformation

- Need to define and create test data, i.e. build class models with various properties, e.g.:
  - with(out) inheritance
  - with one persistent class
  - with multiple persistent classes
  - etc...
- Need to define an oracle that validates that the produced table is correct for each input test data, e.g.:
  - only 1 persistent class A in the input gives exactly 1 table called A in the output model
- The oracle needs to manipulate the two models and check properties
- The expected properties need to be determined for each test case

Validate the output models

- Checks the validity of the output model returned by the transformation of one test model
- Often this is considered to be a model comparison problem (i.e. expected outputs are known)
- Producing expected outputs for all test cases can be difficult and error-prone
- The oracle can be considered as a parameterized function with parameters as follows:
    1. The output model returned by the transformation
    2. "Oracle data" provided by the tester. e.g. the expected model or the input test model

Types of oracle functions

Model comparison	Output model is compared directly to an expected output model.
Contracts	Pre- and post-conditions are applied to the models before and after the transformation.
Pattern matching	Look for the presence of specific patterns within the model, either using OCL assertions or model snippets.

Specific oracle functions

Model comparison

reference model transformation Compare the output model with one from a reference model transformation. The tester to provide the reference model transformation.

inverse transformation Compare input model with the result of applying the transformation followed by the inverse transformation. The tester must provide an inverse transformation that is guaranteed to produce the same model, which may not exist.

expected output model Compare the output model with an expected output model. The tester must provide the expected output model.

Specific oracle functions
Rule-based verification

generic contract
: Verify the output model based on the input model and a generic contract (constraints that depend on the input model). The tester must provide the generic contract.

OCL assertion
: Verify the output model based on an OCL assertion. The tester must provide the assertion for each input model.

model snippets
: Verify the output model contains one or more model snippets. The tester must provide the snippets for each input model.

Transformations

- The transformation rule for T, as described earlier, is as follows:
  - *Ru* The persistent classes, and only these ones, are transformed into tables with the same names, except if they inherit directly or not from another persistent class.
- For comparison, a second transformation, T' is used
- The transformation rule for T' is as follows:
  - Ru' The persistent classes, and only these ones, are transformed into tables with the same names.

Output models for illustrative example

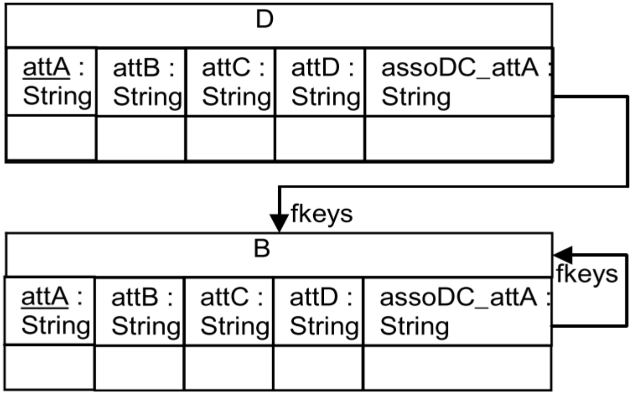

Figure 5: Output model from $T'(Mt1)$

Reference model transformation

- Transformation implemented with Kermeta in 113 lines of code with 11 operations
- Reference transformation implemented with Tefkat in 94 lines of code, 8 patterns, and 5 rules
- Both implementations have similar complexity
- Both implementations have to be maintained as requirements change
- Both implementations may have a steep learning curve
- Maintaining both models is too complex for a tester and must be a developer task

Inverse transformation

- The transformations are not injective
- There is no way to recover class A or C from the output model
- Comparison through inverse transformation is not possible for these transformations

Expected model

- The tester must produce both output models
- Both models are as complex as the input model
- Not possible to simply evolve expected output from T to the expected output from T'
- Maintaining both transformations T and T' requires twice the effort as maintaining one transformation

Generic contract

Contract for this example

```
post table_correctly_created :
result.table.size=inputModel.classifier
   .select(cr|cr.oclIsTypeOf(Class))
   .select(c|c.oclAsType(Class).is_persistent)
   .select(cp|not cp.oclAsType(Class).parents
        .exists(p | p.is_persistent)).size
and //note:  the classes have different names
inputModel.classifier
   .select(cr|cr.oclIsTypeOf(Class))
   .select(c|c.oclAsType(Class).is_persistent)
   .select(cp|not cp.oclAsType(Class).parents
        .exists(p | p.is_persistent))
   .forAll(csp|result.table
        .exists(t |t.name = csp.name))
```

Generic contract

- Contracts can be very complex compared to the textual description of each rule
- Contracts are relatively simple to evolve
- Complexity in the contracts can be difficult to maintain and may hide faults in the specification

Model snippets

Snippets for this example

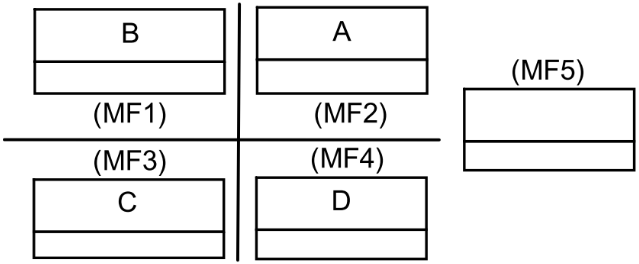

Figure 6: Five *RDBMS* model snippets

```
o1:    (MF1 , 1 , =)
o2:    (MF2 , 0 , =), (MF3 , 0 , =)
o3:    (MF4 , 0 , =)
o4:    (MF5 , 1 , =)
```

Model snippets

- There are 5 model snippets and 4 oracle functions
- $o1$ validates “The persistent classes are transformed into tables with the same names”
- $o2$ validates “and only these ones”
- $o3$ validates “except if they inherit directly or not from another persistent class”
- $o4$ validates 'and only these ones” in a more generic way by checking the total number of tables produced
- Oracle functions are easy to write, modularize, and adapt to a new transformation
- For T', $o3$ and $o4$ each increment cardinality by 1

OCL assertions

- Similar in concept to model snippets
- For example, *o*2 would be written:
  `result.table.select(t|t.name=A).size()=0 and`
  `result.table.select(t|t.name=C).size()=0`
- assertions are slightly less easy to write
- assertions are easy to modularize and adapt to a new transformation

Considering context

- Context is a significant consideration when choosing an oracle function
- Context can differ based on

  complexity of transformation Very complex transformations make reference or inverse model transformations difficult to produce and to maintain.

  complexity of output models Very complex output models are difficult to verify using full model comparison.

  number of test models Very large numbers of test models make overly-specific oracle functions hard to produce and maintain.

  reuse and evolution If the transformation is likely to change over time, generic oracles such as reference or inverse model transformations may not be reusable across versions.

Conclusion

- Motivated the desire to use oracle functions
- Motivated the difficulty of choosing a good oracle function
- Described several types of oracle functions with 6 concrete functions
- Illustrated each concrete oracle function within a single example
- Provided context-based guidance for choosing an oracle function
- Future work: provide strict criteria to measure the strengths and weaknesses of each oracle function within several contexts

Discussion

- How likely do you think it is that an inverse transformation can be used (i.e. is this really a viable option)?
- How many reference implementations do we have to make before we convince ourselves that any one of them is correct?
  - How can we simplify the creation of a reference implementation?
  - Are there any other advantages to creating a reference implementation?
- The paper only makes intuition-based observations within each context. They admit that they need hard metrics.
  - What was the point of the paper if only to give hand-wavey observations?
  - Do you agree with their observations?
  - How might they apply metrics within each context?