Search-based Model Transformation Analysis*

4th Workshop on the Analysis of Model Transformations (AMT) @ MoDELS‘15

Manuel Wimmer
Business Informatics Group
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

*This work is co-funded by the European Commission under the ICT Policy Support Programme, grant no. 317859 (ARTIST) as well as by the Christian Doppler Forschungsgesellschaft and the BMWFW, Austria.
Introduction

- **Model Transformations** are widely used/usable today

- **For general techniques**
  - Model exchange, diffing, merging, versioning, evolution, execution, annotations, verification, modernization, …

- **For specific application areas**
  - DB, OO, Web, documents, Cloud, social networks, production systems, buildings, …

- **To ensure their proper usage, different properties are desirable**
  - Classical properties such as termination, confluence, …
  - Correctness w.r.t. specifications
  -ilities such as readability, extensibility, maintainability, …

---

Analysis of Model Transformation (AMT)

Why?

- Required in **many** different model engineering **processes** and **tasks**

- Our past experiences (excerpt)

![Transformation Diagram]

**Transformation Testing**
- Requirements
- Debugging
- Implementation
- Testing

**Transformation Refactoring**
- Testing
- Smell Detection
- Counter-measures
- Improvements

**Transformation Reuse**
- Abstract
- Integrate
- Find
- Specialize
### Transformation Testing

- **Languages to specify transformation contracts**
- **Languages to implement transformations**
- **Mapping between contracts and rules**
- **Requirements**
- **Debugging**
- **Implementation**
- **Testing**

**Automated testing with OCL and MT engines**

**Thursday Oct 1, 3:30 p.m.**

**Fully Verifying Transformation Contracts for Declarative ATL Foundations**

Bentley James Oakes, Javier Troya, Levi Lúcio, Manuel Wimmer

AMT
How?

- Transformation Refactoring

### Transformation Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoC</td>
<td>30</td>
<td>44</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>@Elements</td>
<td>122</td>
<td>132</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>@Rules</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>@ARules</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>@Collapses</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>@Helpers</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>@Collapses</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>@CDX</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>@CGD</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>MIB</td>
<td>13</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Avg. File-In</td>
<td>0</td>
<td>0.83</td>
<td>0.83</td>
<td></td>
</tr>
<tr>
<td>Avg. Transformed</td>
<td>18</td>
<td>0.17</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>Avg. NoTransformed</td>
<td>17</td>
<td>5.17</td>
<td>3.8</td>
<td></td>
</tr>
<tr>
<td>Avg. Val-In</td>
<td>1</td>
<td>1.89</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Avg. Value</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Abstraction</td>
<td>0</td>
<td>0.33</td>
<td>0.33</td>
<td></td>
</tr>
</tbody>
</table>

### Transformation Refactoring Catalogue

<table>
<thead>
<tr>
<th>Refactoring</th>
<th>QVT-R</th>
<th>QVT-O</th>
<th>ETL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rename</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rename In/Out Pattern Element</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rename In/Out Model</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rename Rule/Order</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Extract Helper/Rule</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Inline Helper/Rule</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Merge Rule</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Split Rule</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Merge Binding</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Split Binding</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Correct Rule Type</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Inheritance-related</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Extract Superrule</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pull Up Binding</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pull Up Filter</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Eliminate Superrule</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Push Down Binding</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Push Down Filter</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

### Refactoring Space Exploration

1. Extract Matched Rule from Distinct-Foreach Output Pattern Element
2. Convert Helper Type from OCL Expr
3. Extract Attribute Helper
4. Extract Lazy Rule from Distinct-Foreach Output Pattern Element
5. Extract Superrule
6. Pull-up binding 2x
7. Pull-up Filter 2x
8. Merge Bindings
9. Extract Lazy Rule from Distinct-Foreach Output Pattern Element
10. Extract Superrule
11. Pull-up binding 2x
12. Pull-up Filter 2x
13. Merge Bindings

### Including Performance Impacts

<table>
<thead>
<tr>
<th>Transformations</th>
<th>Instructions</th>
<th>CPU Time</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>17,125,641</td>
<td>18.07 s</td>
<td>1.00</td>
</tr>
<tr>
<td>T2</td>
<td>13,380,985</td>
<td>15.14 s</td>
<td>1.19</td>
</tr>
<tr>
<td>T3</td>
<td>256,444,024</td>
<td>186.31 s</td>
<td>0.10</td>
</tr>
<tr>
<td>T4</td>
<td>97,064,018</td>
<td>53.20 s</td>
<td>0.34</td>
</tr>
<tr>
<td>T5</td>
<td>12,237,967</td>
<td>9.07 s</td>
<td>1.99</td>
</tr>
<tr>
<td>T6</td>
<td>12,888,967</td>
<td>10.65 s</td>
<td>1.70</td>
</tr>
</tbody>
</table>

AMT
How?

- Transformation Reuse

Reusing Mechanisms for Model Transformations

<table>
<thead>
<tr>
<th>Scope</th>
<th>Specificity</th>
<th>Granularity</th>
<th>Classified Reuse Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>intra</td>
<td>concrete</td>
<td>small</td>
<td>Code Scavenging, User-defined Functions, Rule Inheritance</td>
</tr>
<tr>
<td>intra</td>
<td>concrete</td>
<td>small</td>
<td>Module Import, Transformation Product Lines</td>
</tr>
<tr>
<td>intra</td>
<td>generic</td>
<td>small</td>
<td>HOT, AOP, Reflection, Generic Functions, Embedded DSLs</td>
</tr>
<tr>
<td>intra</td>
<td>generic</td>
<td>large</td>
<td>Generic Transformations, Stand-alone DSLs</td>
</tr>
<tr>
<td>intra</td>
<td>concrete</td>
<td>large</td>
<td>Orchestration</td>
</tr>
</tbody>
</table>

HOT as example for specialization and integration

Populated Transformation Repositories needed!

AMT

When?

- Model Transformation Pattern Revisited

\[ TM = MT(SM) \]

\[ aMT = F(SM, TM, TP) \]
A²MT: Analysis of A Model Transformation

When?

Source Metamodel «conformsTo» Target Metamodel

Source Models «input» aModel Transformation «output» Target Models

Transformation Program «refersTo» Target Metamodel

Source Metamodel «refersTo» Transformation Program

Out-place

Traceability, Consistency

In-place

Evolution
Model Evolution as Transformation Reconstruction
Model Evolution as Transformation Reconstruction

- Basic model evolution problem...

Which modeling operations have occurred?

- … occurs in parallel evolution scenarios

Multi-user

Multi-viewpoint
Operation Detection
Baseline: Atomic Operations

- **Model differencing**
  - **Comparison of states** of an artifact
  - **Match function** to find **correspondence** of two elements in compared artifacts
  - **Differences** are converted into **atomic operations**

- **Atomic operation types**
  - Add model element
  - Delete model element
  - Feature update
    - Insert feature value
    - Delete feature value
    - Feature order change
  - Move
Operation Detection
Extension: Composite Operations

- Better understanding the evolution of a model
  - Going beyond atomic operations
  - Composite operations (e.g., refactorings)
    - More concise description of evolution
    - Reason about the intent(s) of an evolution

- Composite operations are model transformations
  - Preconditions
  - Sequence of atomic operations
  - Postconditions

---

Origin Model | Revised Model
---|---
Bike | Speed: Int | HorsePwr: Int
Car | Speed: Int | HorsePwr: Int

Atomic changes
- Add Class
- Update Class.superClass
- Update Class.superClass
- Move Attribute
- Move Attribute
- Delete Attribute
- Delete Attribute

---

Operation Detection by Model Differencing

- **Two-phase process**
  - **Phase 1**: Matching for finding correspondences between objects
  - **Phase 2**: Fine-grained comparison of corresponding objects for finding atomic differences
  - **Phase 3**: Aggregation of atomic differences for detecting composite operation applications

---

**Legend**

- $M_{V_0 \cdot V_1}$ ... Match Model
- $D_{V_0 \cdot V_1}^{\text{atom}}$ ... Diff Model [atomic]
- $D_{V_0 \cdot V_1}^{\text{comp}}$ ... Diff Model [composite]
Bridging Model Transformation and Model Differencing

Precondition Model
- t1: Class
- t2: Property
- t3: Class
- t4: Property

Diff Pattern
- od1: DiffElement
  - kind=UPDATE
  - feature=superClass
- od2: DiffElement
  - kind=UPDATE
  - feature=superClass
- od3: DiffElement
  - kind=ADD
- od4: DiffElement
  - kind=MOVE
- od5: DiffElement
  - kind=DELETE

Postcondition Model
- t1: Class
- t3: Class
- t5: Class

Extract SuperClass

Initial Model
- Bike
  - speed: Int
  - horsePwr: Int

Diff Model
- o1: Class
  - name="Bike"
  - ownedAttribute

- o2: Property
  - name="speed"

- o3: Property
  - name="horsePwr"

- o4: Class
  - name="Car"

- o8: Class
  - name="Vehicle"

Revised Model
- o1: Class
  - name="Bike"
  - superclass=Vehicle

- o2: Property
  - name="speed"

- o3: Property
  - name="horsePwr"

- o4: Class
  - name="Car"

Preconditions
- Attribute
  - name="horsePwr"

Postconditions
- Attribute
  - name="horsePwr"
Operation Detection Cases (1/2)

- Independent Operations
  - Op₁
  - Op₂

- Overlapping Operations: Type I
  - Op₁
  - Preconditions of Op₂
  → Iterative Forward Detection Approach

- Overlapping Operations: Type II
  - Op₂
  - Postconditions of Op₁
  → Iterative Backward Detection Approach
Operation Detection Cases (2/2)

- Overlapping Operation: Type III (Type I + Type II)
  - Neither Op₁ nor Op₂ entirely detectable
  - Preconditions of Op₁ valid
  - Postconditions of Op₂ valid

![Diagram of overlapping operations](image)
Search-based Operation Detection

- **Huge Search Space**
  - Multiple combinations
  - Order matters
  → Meta-heuristic search-based approach

- **Genetic Algorithm** for Operation Detection

Search-based Operation Detection

Vo

Set of \( \Delta \)

1) Selection
2) Crossover
3) Mutation

V_{c1}

while not reached max iterations

\( \Delta_{min} \rightarrow \) Fitness Function

<table>
<thead>
<tr>
<th>Op1</th>
<th>Op2</th>
<th>Op3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoveAtt(...)</td>
<td>Rename(...)</td>
<td>MoveAtt(...)</td>
</tr>
<tr>
<td>ExtractClass(…)</td>
<td>MoveRef(…)</td>
<td>Rename(…)</td>
</tr>
<tr>
<td>ExtractClass(…)</td>
<td>PullUp(…)</td>
<td>MoveAtt(…)</td>
</tr>
</tbody>
</table>
**Evolution** of the Graphical Modeling Framework (GMF)

- http://www.eclipse.org/modeling/gmp/
- Evolution of three Ecore-based metamodels
- Object-oriented refactoring catalogue
- Independent and overlapping operation occurrences

<table>
<thead>
<tr>
<th>Model</th>
<th># Ops</th>
<th># Elements</th>
<th># Values</th>
<th># Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMF Map</td>
<td>8</td>
<td>367,428</td>
<td>620,735</td>
<td>683,784</td>
</tr>
<tr>
<td>GMF Graph</td>
<td>24</td>
<td>277,310</td>
<td>521,588</td>
<td>629,695</td>
</tr>
<tr>
<td>GMF Gen</td>
<td>93</td>
<td>883,1295</td>
<td>1385,2188</td>
<td>1935,2899</td>
</tr>
</tbody>
</table>

**Results**

<table>
<thead>
<tr>
<th>Model</th>
<th>Matching Approach</th>
<th>Genetic Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision Recall</td>
<td>Precision Recall</td>
</tr>
<tr>
<td>GMF Map</td>
<td>100% 100%</td>
<td>100% 85%</td>
</tr>
<tr>
<td>GMF Graph</td>
<td>100% 50%</td>
<td>95% 87%</td>
</tr>
<tr>
<td>GMF Gen</td>
<td>100% 73%</td>
<td>94% 97%</td>
</tr>
</tbody>
</table>
Model Transformation as Optimization Problem
Model Transformation as Optimization Problem

- From Search-based Operation Detection…

  - Initial Model
  - Revised Model

  Genetic Algorithm

  Operation Types

  Operation Sequence

- …to Search-based Model Transformations

  - Initial Model
  - Objectives

  Meta-Heuristic Search

  Transformation Rules

  Rule App Sequence(s)

M. Kessentini, P. Langer, M. Wimmer: Searching Models, Modeling Search: On the Synergies of SBSE and MDE; CMSBSE@ICSE 2013, pp. 51-54.
Aim 1: Explicating Transformation Goals

What are the goals of such transformations?
Aim 2: Guided Exploration of Transformation Space

How can I find potential solutions without exhaustively exploring everything?
Motivating Example: Modularization Case Study

- mtunis (operating system for educational purposes)

Modularization Case Study

- Represent problem domain through metamodel

- Represent modularization through graph transformation rules
Modularization Case Study

- Goal: find a good OO module design

- Objectives: follow an Equal-Size Cluster Approach (ECA)
  - Coupling : min
  - Cohesion : max
  - Modularization quality (MQ) : max
  - Number of modules : max
  - Difference between max and min number of classes in a module : min

- Complexity corresponds to Bell number
  - \[ B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k \]
  - mtunis example: 20 classes → 51724158235372 possible solutions
Modularization Case Study

- How can we achieve the objectives?
  - It is not trivial to find a good rule order
  - Rule parameters might have an infinite value range
  - Producing each possible output model is expensive
  - Evaluating each possible output model is expensive as well

- Is there any way of automating the search for good rule orchestrations?

  → Search-based software engineering (SBSE) [1] to the rescue!
    - Search-based Optimization using Meta-heuristics
    - Many software engineering problems have been solved in this way

- But how may a **generic**, **transparent**, **declarative**, and **guided** approach look like?

---

**MOMoT – Marrying Search-based Optimization and Model Transformation Technology**

- **Bridging two worlds**
  - MDE to model problem domains and create problem instances and model transformations to manipulate problem instances
  - SBSE techniques to efficiently handle potentially infinite search spaces

→ **Focus on reusing existing technologies and loose coupling!**
  - Henshin Graph Transformation Engine (http://eclipse.org/henshin)
  - Extended MOEA Framework (http://moeaframework.org)

---

MOMoT – Objectives

- Objectives specify goals of a transformation
  - **Domain-specific** objectives are related to the problem domain
    - Example: Minimize coupling between modules
  - **Solution-specific** objectives are related to the solution representation
    - Example: Minimize the number of rule applications of a solution

- Provide objectives in **SCL** based on **OCL** and **Java**

```java
objectives = { // auto-inject the solution and the root of the model
  Coupling : minimize { // Java-like syntax, access attribute storage
    solution.getAttribute("calculation", typeof(Calculator)).metrics.coupling
  }
  Cohesion : maximize {
    solution.getAttribute("calculation", typeof(Calculator)).metrics.cohesion
  }
  MQ : maximize {
    solution.getAttribute("calculation", typeof(Calculator)).metrics.mq
  }
  MinMaxDiff : minimize {
    solution.getAttribute("calculation", typeof(Calculator)).metrics.mmDiff
  }
  NrModules : maximize "self.modules->size()" // OCL-specification
  Length : minimize new TransformationLengthDimension // generic objective
}
```
MOMoT – Constraints

- Constraints define the validity of solutions
  - **Domain-specific** constraints are related to the problem domain
    - Example: There should not be empty modules
  - **Solution-specific** constraints are related to the solution representation
    - Example: The number of rule applications of a solution must be less than 20

- Provide constraints in **SCL** based on **OCL**, **Java**, and **Graph Patterns**

```plaintext
constraints = { // mark invalid solutions
  UnassignedFeatures : minimize { // Java-like syntax, direct calculation
    (root as ModularizationModel).features.filter[c | c.module == null].size
  }
  EmptyModules : minimize {
    (root as ModularizationModel).features.filter[m | m.classes.empty].size
  }
}
```
MOMoT – Exploration Configuration

- Exploration Configuration
  - MOMoT is task- and search algorithm-agnostic
  - Reuse existing search algorithms and their possible instantiations
    - Population-based search methods (provided by MOEA)
    - Local Search methods (extensions to MOEA)
  - Provide concrete instantiations of algorithms in SCL

```java
algorithms = {  // moea, local, and configuration are auto-injected objects
  NSGAIII : moea.createNSGAII(  
    0, 6, // inner and outer divisions
    new TournamentSelection(2), // k == 2  
    new OnePointCrossover(1.0), // 1.0 == 100%, always do crossover  
    new TransformationPlaceholderMutation(0.15), // 15% mutation  
    new TransformationVariableMutation(orchestration.searchHelper, 0.10))  
  HillClimbing : local.createHillClimbing( // at most 100 neighbors  
    new IncreasingNeighborhoodFunction(configuration.searchHelper, 100),  
    new ObjectiveFitnessComparator("MQ"))  
  RandomSearch : new RandomSearch( // without auto-injected objects  
    configuration.problem, // problem instance  
    configuration.createPopulationGenerator(300), // random solutions  
    new NondominatedPopulation) // population class to use
}```
MOMoT – Exploration

- Population-based Search methods
  - Maintains a set of candidate solutions at once (population)
  - Uses different operators to manipulate the population
    - Selection: Selecting good solutions for recombination
    - Crossover: Recombine two existing solutions
    - Mutation: Introduce slight changes into solutions to explore different areas

Examples: ε-MOEA, NSGA-II, NSGA-III, etc.
MOMoT – Exploration

- Local Search methods
  - Maintains one solution at a time
  - Generate neighbor solutions using a neighborhood function
    - Neighbors only differ slightly from the original solution
  - Move to next solution depending on the found neighbors and their fitness

- Examples: Random Descent, Hill Climbing, Simulated Annealing, etc.
Modularization Case Study: MOMoT Results

- Example solution computed by MOMoT
  - Using NSGA-III
  - Solution length of 50
  - Population size of 300
Modularization Case Study: MOMoT Results

- Runtime performance
  - Generic MDE encoding vs dedicated, native encoding
  - Average runtime for each encoding
  - MOMoT on average slower by a factor of 3-15 / 10-13
Conclusion & Outlook
Conclusion

- New research directions
  - Usage of search-based techniques for transformation analysis
  - Analyze and guide individual transformation executions

- Several application cases
  - A-priori
    - Search-based approaches have been applied for transformation testing
    - Ongoing work: Search-based transformation NFP improvement
  - On-the-fly
    - Using objectives to guide the transformation execution
  - A-posteriori
    - Reconstruct a transformation
    - Reason about different explanations of possible transformations
    - Generate a transformation program (MTBE)
  - …
Outlook

- **Reproduction studies** for evaluating MOMoT
  - **Model Matching**: Precise and complete match model
  - **Model Diffing**: Short paths explaining the complete evolution
  - **Model Merging**: Apply as many operations as possible while ensuring general constraints

- **Comparative studies** with other emerging search-based transformation approaches
  - Model search algorithms as model transformations [1]
  - Enhance transformation engine with specific search algorithm [2]
  - Compile models to typical search encoding representation [3]

---


Thank you!
Comments? Questions? Feedback?

Manuel Wimmer
wimmer@big.tuwien.ac.at

MOMoT on github
http://martin-fleck.github.io/momot