A Survey on
Incremental Model Transformation Approaches

Models and Evolution Workshop, Miami, Florida, Sept. 30th 2013


Dept. of Cooperative Information Systems
Johannes Kepler University Linz
http://www.cis.jku.at

Business Informatics Group
Vienna University of Technology
http://www.big.tuwien.ac.at

LieberLieber Software GmbH
http://www.lieberlieber.com

Funding: BRIDGE 832160
Need for Incremental Model Transformations

Motivation • Language Coverage • Execution Phases • Overhead • Lessons Learned

- **Model transformations** allow to generate models out of other models
- **Evolution entails re-execution** of model transformations
- **Batch transformations** generate complete target models each time the transformation is (re-)executed
- In contrast, **incremental model transformations** propagate changes in the source model, only
  - Rather depend on size of changes, than on size of the source model
  - Especially useful for frequent changes in large source models

Recent approaches for **incremental model transformations** differ substantially
Motivating Example

**Motivation**
- Language Coverage
- Execution Phases
- Overhead
- Lessons Learned

---

**Language Coverage**

**Execution Phases**

**Overhead**

**Lessons Learned**

---

Which language parts are supported for incremental transformation?

How is incrementality achieved?

Change in the source model

Change is propagated to target model

How much overhead is imposed by incrementality?
Evaluation Framework Overview

Motivation ■ Language Coverage ■ Execution Phases ■ Overhead ■ Lessons Learned

<table>
<thead>
<tr>
<th>Approach</th>
<th>Giese et al., 2008</th>
<th>Lauder et al., 2012</th>
<th>Hearnden et al., 2006</th>
<th>Jouault and Tisi, 2010</th>
<th>Rath et al., 2008</th>
<th>Bergmann et al., 2012a</th>
<th>Bergmann et al., 2012b</th>
<th>Razavi et al., 2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Coverage</td>
<td>TGG</td>
<td>Viatra2</td>
<td>Viatra2</td>
<td>Viatra2</td>
<td>Viatra2</td>
<td>Viatra2</td>
<td>Arbitrary</td>
<td></td>
</tr>
</tbody>
</table>

Which language parts are supported for incremental transformation?

- Declarative Language Parts
- Imperative Language Parts

How is incrementality achieved?

- Change Detection
- Impact Analysis
- Change Propagation

How much overhead is imposed by incrementality?

- Specification
- Run-time
- Memory

8 approaches have been examined.
### Motivation

- **Language Coverage**
- **Execution Phases**
- **Overhead**
- **Lessons Learned**

### Declarative Language Parts

- **Number of Input Elements**
  - Giese, 2008: TGG
  - Lauder, 2012: TGG
  - Hearnden, 2006: Tefkat
  - Jouault, 2010: ATL
  - Rath, 2008: Viatra2
  - Bergmann, 2012a: Viatra2
  - Bergmann, 2012b: Viatra2
  - Razavi, 2009: Arbitrary

- **Number of Output Elements**
  - 1..*

- **Assignments**
  - ✓

- **Rule Inheritance**
  - -

- **Conditions**
  - ✓

### Imperative Language Parts

- **Number of Input Elements**
  - 1..*

- **Number of Output Elements**
  - 1

- **Assignments**
  - 1..*

- **Rule Inheritance**
  - -

- **Conditions**
  - ✓

### Insufficient Support for Imperative Language Parts

- **Giese, 2008**
- **Lauder, 2012**
- **Hearnden, 2006**
- **Jouault, 2010**
- **Rath, 2008**
- **Bergmann, 2012a**
- **Bergmann, 2012b**
- **Razavi, 2009**

---

### ATL Code

```plaintext
to t: Relational!Table (  
  col <- c.attr->select(e | not e.multiValued)  
  ->subSequence(1,2)  
) do {  
  thisModule.id <- thisModule.id + 1; } 

helper def : id : Integer = 0;  
abstract rule NamedEl2Named {  
  from n : ClassD!NamedElement  
  to t : Relational!Named (  
    name <- n.name  
  )  
}  
rule Class2Table extends NamedEl2Named {  
  from c : ClassD!Class (c.persistent)  
  to t: Relational!Table (  
    col <- c.attr->select(e | not e.multiValued)  
    ->subSequence(1,2)  
  )  
  do {  
    thisModule.id <- thisModule.id + 1;  
  }  
}
```

---

### Reference

- Giese, 2008
- Lauder, 2012
- Hearnden, 2006
- Jouault, 2010
- Rath, 2008
- Bergmann, 2012a
- Bergmann, 2012b
- Razavi, 2009
Evaluation Framework

Motivation | Language Coverage | Execution Phases | Overhead | Lessons Learned

Language Coverage
- Declarative Language Parts
- Imperative Language Parts

Overhead
- Specification
- Run-time
- Memory

Execution Phases
- Change Detection
- Impact Analysis
- Change Propagation
Change Detection

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Minimality</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Type &amp; Number of Changes</td>
<td>1..*</td>
<td>1..*</td>
<td>1</td>
<td>1</td>
<td>1..*</td>
<td>1..*</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Insert</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Delete</td>
<td>1..*</td>
<td>1..*</td>
<td>1</td>
<td>1</td>
<td>1..*</td>
<td>1..*</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Update</td>
<td>1..*</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1..*</td>
<td>1..*</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Move</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Granularity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Object</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Value</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Link</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Change Log Optimization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Focus on Basic Change Operations

(i) Compare to previous version of model to detect changes
(ii) Listen for changes

(i) Detect single changes
(ii) Detect multiple changes
(i) Detect on course-grained level
(ii) Detect on fine-grained level
Impact Analysis

Motivation
Language Coverage
Execution Phases
Overhead
Lessons Learned

Impact Analysis
- Required Knowledge of Transformation Specification
  - TGG
  - TGG
  - ATL
  - Viatra2
  - Viatra2
  - Viatra2
  - Arbitrary
- Black Box
- White Box
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
- Trace
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
- Model2Model
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
- Model2Transformation
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
- Compile-Time
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
- Run-Time
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
- Auxiliary Information
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓

Trace Information is Mandatory

• Required Knowledge of Transformation Specification
• Black Box
• White Box
• Trace
• Model2Model
• Model2Transformation
• Compile-Time
• Run-Time
• Auxiliary Information

Impact Analysis

Source Model

c1: Class
name="Person"
persistent=true

a1 : Attribute
name="Fullname"
multivalued=false

d1 : DataType
name="String"
a2 : Attribute
name="Address"
multivalued=false

Target Model

t1: Table
name="Person"

Model2Transformation Trace

TraceLink
TraceLink
TraceLink

Model2Model Trace

TraceLink
TraceLink
TraceLink

Execution Engine

e.g.

- Compiler-generated rules
- Database tables
Change Propagation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TGG</td>
<td>TGG</td>
<td>Tefkat</td>
<td>ATL</td>
<td>Viatra2</td>
<td>Viatra2</td>
<td>Viatra2</td>
<td></td>
<td>Arbitrary</td>
</tr>
<tr>
<td>• Time</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Eager (Synchron)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Lazy (Asynchron)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>• Coverage</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Partial</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>• Complete</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Granularity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Coarse (Rule)</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Fine (Binding)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n.a.</td>
</tr>
<tr>
<td>• Direction</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>• Unidirectional</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Bidirectional</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Strategy</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Lack of Appropriate Propagation Strategies

(i) Re-execute complete rule
(ii) Re-execute bindings only

(i) Support different strategies
(ii) Automatically select strategies

Source Model

Target Model

Execution Engine

(i) Support different strategies
(ii) Automatically select strategies
Evaluation Framework

- Declarative Language Parts
- Imperative Language Parts

Language Coverage

Evaluation Framework

Overhead
- Specification
- Run-time
- Memory

Execution Phases
- Change Detection
- Impact Analysis
- Change Propagation

Motivation ■ Language Coverage ■ Execution Phases ■ Overhead ■ Lessons Learned
Overhead

<table>
<thead>
<tr>
<th>Language Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Giese, 2008</td>
</tr>
<tr>
<td>Lauder, 2012</td>
</tr>
<tr>
<td>Hearnden, 2006</td>
</tr>
<tr>
<td>Jouault, 2010</td>
</tr>
<tr>
<td>Rath, 2008</td>
</tr>
<tr>
<td>Bergmann, 2012a</td>
</tr>
<tr>
<td>Bergmann, 2012b</td>
</tr>
<tr>
<td>Razavi, 2009</td>
</tr>
</tbody>
</table>

- **No Specification Overhead**
  - TGG
  - TGG
  - Tefkat
  - ATL
  - Viatra2
  - Viatra2
  - Viatra2
  - Arbitrary

- **No Run-time Overhead**
  - ✓
  - ✓
  - unknown
  - unknown
  - unknown
  - ✓
  - unknown
  - unknown

- **No Memory Overhead**
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓
  - ✓

**Motivation**

**Execution Phases**

**Overhead**

(i) **No Specification Overhead**
(ii) **Annotations**
(iii) **New Specification**

(i) **Model2Model Trace**
(ii) **Model2Transformation Trace**
(iii) **Auxiliary Information**

(i) **Generation of Trace Information**
(ii) **Maintaining Trace Information**

**Trade-off between space and time**
Lessons Learned

Motivation | Language Coverage | Execution Phases | Overhead | Lessons Learned

Language Coverage

- Declarative Language Parts
- Imperative Language Parts

Evaluation Framework

- Specification
- Run-time
- Memory

Overhead

- Insufficient Support for Imperative Parts

Execution Phases

- Focus on Basic Change Operations
- Change Detection
- Impact Analysis
- Change Propagation

Lessons Learned

- Trace Information is Mandatory
- Lack of Proof of Correctness
- Lack of Appropriate Propagation Strategies
References