Fact or Fiction – Reuse in Model-to-Model Transformations*


5th International Conference on Model Transformation

Johannes Schönböck
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 has been partly funded by the Austrian Science Fund (FWF) under grant P21374-N13.
Motivation

- Model transformations are the **heart and soul** of MDE

- Model transformations are mostly developed **from scratch**

- For increasing **development productivity** and **quality** → reuse mechanisms are indispensable

- **Numerous reuse mechanisms** have been proposed → however, **no systematic comparison** exists

- **In-depth comparison** of proposed reuse mechanisms needed to
  - identify their application scenarios and how to apply them
Comparison of Reuse Mechanisms – Scope 2

1. How to reuse transformation logic within a single transformation?

2. How to reuse existing transformation logic in similar scenarios (same MMs, different logic)?

3. How to reuse existing transformations in different scenarios (different MMs, same logic)?

4. How to reuse cross-cutting (scenario-independent) transformation logic?

5. How to reuse transformations in the large?
Comparison Criteria

1. How to abstract from details, such that an artifact might be reused?

Abstraction
- Generalization
  - From Metamodel
  - From Transformation Language
- Simplification
  - Hidden Parts
  - Visible Parts

2. How to efficiently find a reusable artifact?

Selection
- Repository
- Metainformation
- Automatism

3. How to adapt a reusable artifact?

Specialization
- Required Knowledge
- Mechanism
- Language-inherence

4. How to integrate a reusable artifact?

Integration
- Ability
- Kind

Comparison of Reuse Mechanisms – Scope 1

1. How to reuse transformation logic within a **single transformation**?

2. How to reuse existing transformation logic in **similar scenarios** (same MMs, different logic)?

3. How to reuse existing transformations in **different scenarios** (different MMs, same logic)?

4. How to reuse **cross-cutting** (scenario-independent) transformation logic?

5. How to reuse transformations in the **large**? 
Comparison of Reuse Mechanisms – Scope 1

- Functions
- Inheritance*

---

Comparison of Reuse Mechanisms – Scope 2

1. How to reuse transformation logic within a single transformation?

Class2Relational

Class Metamodel

NamedElement
name : String

ClassDiagram
nsURI : String

Class
id : int

0..* classes

Property
type : String

0..* properties

Class2ER
OneTablePerClass

Debugging
Tracing

Class2ER
OneTablePerHierarchy

Debugging
Tracing

Entity Relationship Metamodel

ModelElement
name : String

ERDiagram
URI : String

0..* entities

Entity
id : int

0..* attributes

Attribute
type

Type

ER2Relational

Debugging
Tracing

Relational Metamodel

NamedElement
name : String

Database
table

0..* tables

Column
type

0..* cols

Type

length : int

XML Metamodel

MElement
name : String

Ontology
nSpace : String

OWLClass
id : int

0..* classes

DatatypeProperty
type : String

0..* properties

Ontology2XML

Debugging
Tracing

MEElement
name : String

XMLSchema
nSpace : String

extensionBase

0..* elements

Element
id : int

0..* attributes

Attribute
type : String

2. How to reuse existing transformation logic in similar scenarios (same MMs, different logic)?

3. How to reuse existing transformations in different scenarios (different MMs, same logic)?

4. How to reuse cross-cutting (scenario-independent) transformation logic?

5. How to reuse transformations in the large?
Comparison of Reuse Mechanisms – Scope 2

- **Superimposition**
- Transformation Product Lines

---

**Reusable Artifact**

\[
\text{rule } \text{CDModel2ERModel}\{\ldots}\text{\rule{0.6em}{0.4pt}}
\text{\rule{0.6em}{0.4pt}}
\text{rule } \text{Class2Entity}\{\ldots}\text{\rule{0.6em}{0.4pt}}
\text{\rule{0.6em}{0.4pt}}
\text{rule } \text{Property2Attribute}\{\ldots}\text{\rule{0.6em}{0.4pt}}
\]

**Class2Table_OneTablePerClass**

\[
\text{rule } \text{Class2Entity}\{\ldots}\text{\rule{0.6em}{0.4pt}}
\text{helper def : } \text{Closure}(\ldots)\text{ : }
\text{\rule{0.6em}{0.4pt}}
\text{Set(Property)}
\]

**Class2Table_OneTablePerHierarchy**

Comparison of Reuse Mechanisms – Scope 2

- **Superimposition**
- **Transformation Product Lines**

**Abstraction**
- Generalization
  - From Metamodel
  - From Transformation Language
- Simplification
  - Hidden Parts
  - Visible Parts

**Selection**
- Repository
- Metainformation
- Automatism

**Specialization**
- Required Knowledge
- Mechanism
- Language-inherence

**Implementation**
- Rule Interface

**Integration**
- Ability
- Kind

**Composition**

**Extension**

**Rule Interface**

**ATL Model Transformation Zoo**
- Documentation
- Manual

**Black-box**

**Redefinition**

**Lessons Learned**

**ATL Model Transformation Zoo**

**Motivation**

**Framework**

**Criteria**

**Comparison**

**Lessons Learned**
Comparison of Reuse Mechanisms – Scope 3

1. How to reuse transformation logic within a **single transformation**?

2. How to reuse existing transformation logic in **similar scenarios** (same MMs, different logic)?

3. How to reuse existing transformations in **different scenarios** (different MMs, same logic)?

4. How to reuse **cross-cutting** (scenario-independent) transformation logic?

5. How to reuse transformations in **the large**?
Comparison of Reuse Mechanisms – Scope 3

- **Genericity**
- **Domain Specific Languages**

```java
rule Property2Attribute {
    from property : CD!Property
    to attribute : ER!Attribute (  
        name <- property.name + '_translated',  
        type <- aType  
    ),  
    aType : ER!Type(  
        ...  
    )
}
```
Comparison of Reuse Mechanisms – Scope 3

- Genericity
- Domain Specific Languages*

Comparison of Reuse Mechanisms – Scope 4

1. How to reuse transformation logic within a single transformation?

2. How to reuse existing transformation logic in similar scenarios (same MMs, different logic)?

3. How to reuse existing transformations in different scenarios (different MMs, same logic)?

4. How to reuse cross-cutting (scenario-independent) transformation logic?

5. How to reuse transformations in the large?
Comparison of Reuse Mechanisms – Scope 4

- Higher-Order Transformations*
- Aspect-Orientation
- Reflection

Higher-Order Transformation

rule addDebugMessage{
    from oldAssignment : ATL!Binding
    to assignmentWithDebug : ATL!Binding (propertyName <- oldAssignment.propertyName, value <- debugger ),
    debugger : ATL!OperationCallExp (source <- oldAssignment.value, operationName <- 'debug', arguments <- Sequence {arg} ),
    arg : ATL!StringExp (stringSymbol <- (oldAssignment.outPatternElement.outPattern."rule".name + '.' + oldAssignment.outPatternElement.varName + '.' + oldAssignment.propertyName)
    )
}

reuse Artifact

rule CDModel2ERModel {
    from cdmodel : CD!ClassDiagram
to ermodel : ER!ERDiagram (URI <- cdmodel.nsURI, entities <- cdmodel.classes )
}

Class2ER

URI <- cdmodel.nsURI .debug(CDModel2ERModel.ermmodel.URI),
entities <- cdmodel.classes .debug(CDModel2ERModel.ermmodel.entities)

Comparison of Reuse Mechanisms – Scope 4

- Higher-Order Transformations
- Aspect-Orientiation
- Reflection
Comparison of Reuse Mechanisms – Scope 5

- Orchestration*

Abstraction
- Generalization
  - From Metamodel
  - From Transformation Language
- Simplification
  - Hidden Parts
  - Visible Parts

Implementation
Signature

Integration
- Ability
- Kind

Composition
Coordination

Selection
- Repository
- Metainformation
- Automatism

ATL Model Transformation Zoo
Documentation
Manual

Specialization
- Required Knowledge
- Mechanism
- Language-inherence

Motivation
Framework
Criteria
Comparison
Lessons Learned

## Barriers to Reuse (Lessons Learned)

<table>
<thead>
<tr>
<th>Framework</th>
<th>Motivation</th>
<th>Criteria</th>
<th>Comparison</th>
<th>Lessons Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Insufficient Abstraction from Metamodels and Platform**
- **Missing Repositories for Selection**
- **Lack of Meta-information in Selection**
- **Challenging Specialization Mechanisms**
- **Insufficient Support for Integration in the Large**
Reuse in Model-to-Model Transformations

Fact or Fiction

Thank you for your attention!

http://www.modeltransformation.net