On some Complementary Trends in Model Transformation Generation
Marianne Huchard, Xavier Dolques, Jean-Rémy Falleri, Clémentine Nebut

To cite this version:

HAL Id: lirmm-00534895
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00534895
Submitted on 10 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
On some complementary trends in
Model transformation generation

Marianne Huchard

Joint work with Xavier Dolques, Jean-Rémy Falleri and Clémentine Nebut

LIRMM - University Montpellier 2, CNRS

FTMDD, June 2010
1. MDE/MT/MTG
2. Metamodel alignment based MTG
3. Example based MTG
4. Towards a global MTG architecture
Outline

1. MDE/MT/MTG
2. Metamodel alignment based MTG
3. Example based MTG
4. Towards a global MTG architecture
## Development paradigm
- model-centered

## Advantages
- capitalizing on modelling
- interoperability
- coding technology independent
Consequences

- dependent from modelling technology
- a lot of models, meta-models
- a lot of transformations
The nature of transformations

A few examples

- CIM-PIM-PSM and variants
- software migration
- metamodel version changes
- model building, merging, refactoring

Classifications

# Programming a model transformation

## Actors
- domain expert
- transformation developer

## Languages
- generalist programming languages + model manipulation frameworks (e.g. Java + EMF)
- dedicated programming languages (e.g. QVT, ATL, Kermeta, VIATRA, etc.)

## Required knowledge
- transformation language
- source and target meta-model
- meta-meta-model
- complete specification of the transformation
# The need for generating model transformations

## Context
- Many tools that manipulate models and need to exchange them (code generators, model transformation editors, graphical editors)
- Many evolution of software with technology change
- Many close models (e.g. class models UML, MOF, EMOF, KMT3)
- Many versions of the same metamodel (e.g. UML)

## Support for transformation developers
Automatically generate part of the transformation program
Opportunities for generating model transformations

What makes it possible

- Simplicity of many transformations
- Declarative paradigm (rules: model pattern $\rightarrow$ model pattern)

Close problematics with experience

- Web semantic, ontology alignment, schema matching techniques
- Database, interoperability (ETL tools)

Currently two main tracks

- Metamodel alignment based MTG
- Example (Model) based MTG
Metamodel alignment based MTG

UML metamodel to Entity-Relationship metamodel
Model alignment based MTG

Models (examples)

Rules

UML metamodel to Entity-Relationship metamodel
What we know to do?

### Starting from metamodels
- Metamodel alignment
- Derive rules from alignment

### Starting from models (transformation examples)
- Model alignment
- Derive rules from alignment
Outline

1. MDE/MT/MTG
2. Metamodel alignment based MTG
3. Example based MTG
4. Towards a global MTG architecture
Metamodel alignment (A task in MTG)

Principle

Establishing a match between the two metamodels
# Metamodel alignment (A task in MTG)

## Context
- metamodels: describing same sort of things
  - (class metamodels, traceability metamodels, etc.)

## Interest
- not necessary to have examples (except for testing)
- abstract language manipulation
- prior specification of the transformation is not required
What we did

**Similarity Flooding (Melnik et al.) for matching**
- Similarity flooding works on labeled directed graphs
- Similarity flooding is easily tunable

**Using matching**
- Testing several configurations for Similarity Flooding use
- Definition of a metamodel alignment
- Automatic construction of alignment models
Three steps

The three steps

- From metamodels to graphs
- Application of Similarity Flooding
- Construction of an alignment metamodel using the result of Similarity Flooding
1. From metamodels to graphs
## Transform a metamodel into a labelled directed graph

### Input
- A metamodel

### Output
- A directed labelled graph representing the model

### Objective
- Study the impact on Similarity Flooding of the configuration choice
- Six tested configurations
- Comparison of the results
Configuration Minimal

- Metamodel elements are converted into labelled nodes
- Relations are converted into labelled edges
- Derived attributes, references, operations and parameters are ignored
Next configurations

- **Basic**: separate elements and their names
- **Standard**: adding metaclasses, cardinality and containment
- **Full**: adding derived attributes and references
- **Saturated**: close *supertype*, apply *inheritance*
- **Flattened**: abstract class nodes and *supertype* edges are removed
2. Similarity Flooding
First step: The compatibility graph
First step: The compatibility graph

![Diagram showing the compatibility graph between elements such as `Operation`, `JTypedElement`, `Class`, `JClass`, `NamedElement`, `JElement`, `EString`, `String`, `name`, `name`, `Operations`, `methods`, `type`, `supertype`, `ref`, `type`, `own`, `datatype`, and `type` relationships.]

- `Operation`, `JTypedElement` connected by `ref`.
- `Class`, `JClass` connected by `supertype`.
- `NamedElement`, `JElement` connected by `own`.
- `EString`, `String` connected by `datatype`.
- `name`, `name` connected by `type`.
- `operations`, `type` connected by `ref`.
- `type`, `methods` connected by `ref`.
- `type`, `methods` connected by `supertype`.
- `type`, `type` connected by `supertype`.

Diagram includes additional elements such as `type`, `supertype`, `ref`, and `own` relationships.
Second step: propagation graph

![Propagation Graph Diagram]
Third step: assigning initial similarity values

- 0 if $x$ or $y$ is an identifier (not a model element name)
- $1 - \text{levenshtein}(x, y) / \max(\text{length}(x), \text{length}(y))$ otherwise

<table>
<thead>
<tr>
<th>Compatibility node</th>
<th>Initial similarity value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(NamedElement, JElement)</td>
<td>0.5833334</td>
</tr>
<tr>
<td>(name,name)</td>
<td>1.0</td>
</tr>
<tr>
<td>(EString,String)</td>
<td>0.85714287</td>
</tr>
<tr>
<td>(NamedElement, JTypedElement)</td>
<td>0.6923077</td>
</tr>
<tr>
<td>(Operation, JTypedElement)</td>
<td>0.23076922</td>
</tr>
</tbody>
</table>
Fourth step: propagation and fix point calculus

- **Principle**
  - Propagation of similarity values in the propagation graph, until finding a fix point.
  - Propagation formulae: at step $i$,
    \[ s_{n}^{i+1} = s_{n}^{i} + s_{n}^{0} + \sum_{m \in I^n} w(m, n) \times (s_{m}^{0} + s_{m}^{i}) \]
  - Fixpoint: when similarity values differences is less than $\epsilon$ during two successive steps.
Fifth step : filtering

**Principe**

- To keep best matches.
- A node of $G_{source}$ can match with several nodes of $G_{target}$. 
- A relative similarity value is computed for each node looking at the leaving edge similarities.
- Pairs with a similarity under a threshold are eliminated.
Case study

Objectif

Testing the six configurations

Data

- exMMSource → exMMTarget
- Ecore → Minjava
- Ecore → Kermeta
- Ecore → UML
**Metrics**

Precision, recall et $f$-score :

- **precision** = \( \frac{\text{Number of Correct Found Mappings}}{\text{Number of Total Found Mappings}} \)

- **recall** = \( \frac{\text{Number of Correct Found Mappings}}{\text{Number of Total Existing Mappings}} \)

- **$f$-score** = \( \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \)
Results

- Not so bad results, good precision
- Better results for similar metamodel size
- Configurations Saturated and Basic give good results
Results

- Not so bad results, good precision
- Better results for similar metamodel size
- Configurations Saturated and Basic give good results
Conclusion on metamodel alignment

- A tool that automatically aligns two metamodels
- Assessment of different configurations
- Alignments can be used for the transformation generation, e.g. with the approach of [Lopes et al.]
Outline

1. MDE/MT/MTG
2. Metamodel alignment based MTG
3. Example based MTG
4. Towards a global MTG architecture
# Example based MTG (MTBE)

## Principle
- Inducing transformation rules from transformed models examples.

## Context
- Metamodels: similar to very different;
- A set of examples.

## Interest
- Use of existing data
- Concrete language manipulation
- Prior specification of the transformation is not required
Input data
Output data

Transformation rules

UML

Class

Property +association(Association)

Property −association(Association)

Entity−Relation

Entity

Role

Attribute
Two-step process

Examples

Alignement

Matching ex.

anchor PROMPT

Matching ex.

Rules discovery

Relational Concept Analysis

Rules

Example based MTG Towards a global MTG architecture
Anchor discovery

Anchor pair
An element in a source model which is surely connected to an element of the target model

Hypothesis
When the model is transformed, names remain quite the same

String matching operations
- equality
- substring
- levenshtein (editing) distance
## Principle

- Inspired by anchorPROMPT approach (noy et al.)
- Align a path in the source model and a path in the target model
- Admit a little size difference between the two paths
- Give weights to matchings, then filter
Anchor-based matching process

Original anchorPROMPT propagation
Extension to paths with different size
e.g. generalization in UML versus is-a relation in ER
Precision on case study

number of relevant retrieved matches / number of retrieved matches

![Bar chart showing precision for different cases]

- delegation1
- hideDelegate
- uml-er3
- Families2Persons
- uml-er
- uml-er2
- delegation2
- associations-persons
- emf2km3

- red: precision substring
- green: precision anchorPrompt
Recall on case study

number of relevant retrieved matches / number of relevant matches

![Graph showing recall for case study](image)
Fscore on case study

\[ Fscore = 2 \frac{precision \cdot recall}{precision + recall} \] (best is 1)
Rule discovery

Examples

Rules discovery

Relational Concept Analysis

Rules

- **UML**
  - **Class**
  - **Property** +association(Association)
  - **Property** −association(Association)

- **Entity–Relation**
  - **Entity**
  - **Role**
  - **Attribute**
## Rules discovery

### Discovery process’ properties
- classification of models elements
- classification of mapping links
- derive rules

### Relational Concept Analysis [Huchard et al. 2007]
- extension of Formal Concept Analysis [Wille1982]
- considers relationships in the classification process
Example data

UML model example (seen as instance of the metamodel)

Simplified UML meta-model
Classification of model elements

<table>
<thead>
<tr>
<th>meta-class</th>
<th>Class</th>
<th>Property</th>
<th>Association</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>owner</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>owned</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>owns</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>numero</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Model element classification using their meta-Classes
Classification of model elements

<table>
<thead>
<tr>
<th>meta-class</th>
<th>Class</th>
<th>Property</th>
<th>Association</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>owner</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>owned</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>owns</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>numero</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

concepts

Model element classification using their meta-Classes
Classification of model elements

```
<table>
<thead>
<tr>
<th>owningClass</th>
<th>Account</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>owner</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>
```

Model elements classification using their target by the relation `owningClass`.
Model elements classification

Lattice of the UML model’s elements
Classification interpretation

Concept 3 description.

Concept 6 description.
## Classification properties

### Classification properties of a model element
- its type
- relations of which it is one end
- types of the elements of which it is associated

### Contexts to create
- **Formal contexts:**
  - model elements context
  - meta-model elements context
- **Relational contexts:**
  - instance relation between model and meta-model
  - relations between elements in the model
### Classification of mapping links

<table>
<thead>
<tr>
<th>linkA</th>
<th>Account</th>
<th>Client</th>
<th>number</th>
<th>name</th>
<th>owns</th>
<th>owner</th>
<th>owned</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L4</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>L9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

**Table:** Relation of mapping links with model source elements

<table>
<thead>
<tr>
<th>linkB</th>
<th>Account</th>
<th>Client</th>
<th>number</th>
<th>name</th>
<th>owns</th>
<th>owner</th>
<th>owned</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L4</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>L9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

**Table:** Relation of mapping links with model target elements.
Mappings lattice
Rules extraction
Tools

- Eclipse Modeling Framework (EMF)
- lattice generation plugin : eRCA
- template generation tool : Acceleo
- declarative model transformation language : ATL
The rule Property Attribute
The rule Property Role
Table: Data obtained from the case study (ATL zoo)

<table>
<thead>
<tr>
<th></th>
<th>F2P ¹</th>
<th>B2D ²</th>
<th>C2R ³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source MetaModel size</td>
<td>4</td>
<td>21</td>
<td>5</td>
</tr>
<tr>
<td>Target MetaModel size</td>
<td>5</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Source Model size</td>
<td>23</td>
<td>48</td>
<td>9</td>
</tr>
<tr>
<td>Target Model size</td>
<td>19</td>
<td>59</td>
<td>15</td>
</tr>
<tr>
<td>Mapping size</td>
<td>28</td>
<td>115</td>
<td>18</td>
</tr>
<tr>
<td>ATL transfo. Number of rules</td>
<td>2</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>ATL transfo. Number of helpers</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Generated transfo. Number of rules</td>
<td>6</td>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>Generated target model size</td>
<td>21</td>
<td>54</td>
<td>16</td>
</tr>
<tr>
<td>Bad generated elements</td>
<td>2</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>Missing elements in generation</td>
<td>0</td>
<td>19</td>
<td>2</td>
</tr>
</tbody>
</table>

¹ : Family2Person – ² : BibTex2DocBook – ³ : Class2Relation
1. MDE/MT/MTG
2. Metamodel alignment based MTG
3. Example based MTG
4. Towards a global MTG architecture
Related work

**MM-based MTG**
- Adapted to similar metamodels
  - Ontology-based, pivot ontology (Roser at al., Kappel et al.)
  - Propagation and complete process (Lopes et al.)

**M-based MTG (MTBE)**
- Adapted when examples are known
  - Guiding the way from concrete to abstract syntax with OCL rules (Wimmer et al.)
  - Inductive logics based (Varró et al.)
  - Optimization approach (Kessentini et al.)
**A road map**

### The solution / a mix of
- Alignment
- Learning
- Domain knowledge, semantics

### Open questions
- Improve alignment techniques for metamodels and models
- Propose alternative learning schemes
- Classifying MT - characterizing suitable MTG methods
- Measuring rule interestingness (e.g. support and lift)
- Propose an integrated approach
- Collaboratively build a benchmark
Links

- Gum (similarity flooding alignment) - http://code.google.com/p/gumm-project
- eRCA - http://code.google.com/p/erca

References