



HAL
open science

On some Complementary Trends in Model Transformation Generation

Marianne Huchard, Xavier Dolques, Jean-Rémy Falleri, Clémentine Nebut

► **To cite this version:**

Marianne Huchard, Xavier Dolques, Jean-Rémy Falleri, Clémentine Nebut. On some Complementary Trends in Model Transformation Generation. FTMDD 2010 @ ICEIS 2010: 2nd International Workshop on Future Trends of Model-Driven Development, Funchal, Madeira, Portugal. lirmm-00534895

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

Model Driven Engineering

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

- K. Czarnecki and S. Helsen. Classification of Model Transformation Approaches. 2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA (2003)
- Tom Mens, Pieter Van Gorp : A Taxonomy of Model Transformation. Electr. Notes Theor. Comput. Sci. 152 : 125-142 (2006)

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)

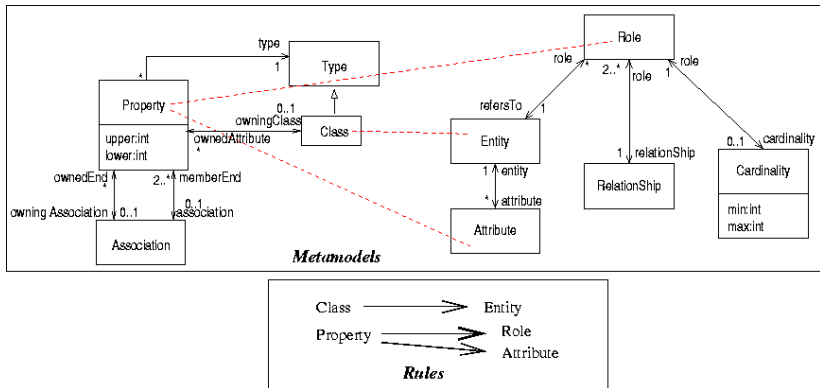
Close problematics with experience

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

Currently two main tracks

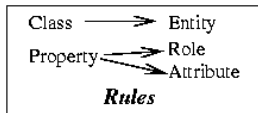
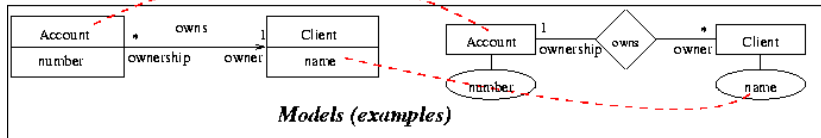
- Metamodel alignment based MTG
- Example (Model) based MTG

Metamodel alignment based MTG



UML metamodel **to** Entity-Relationship metamodel

Model alignment based MTG



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**

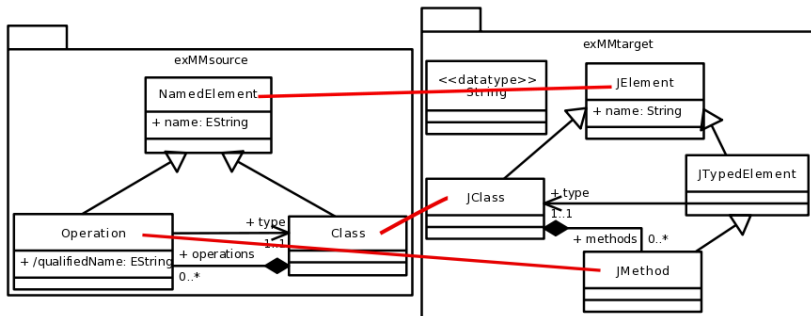
Outline

- 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

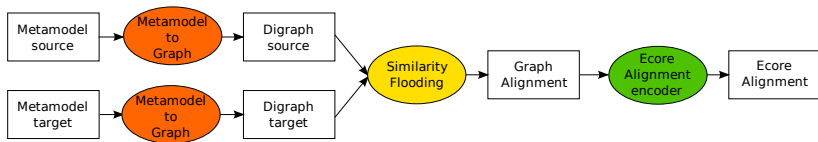
Similarity Flooding (Melnik et al.) for *matching*

- 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

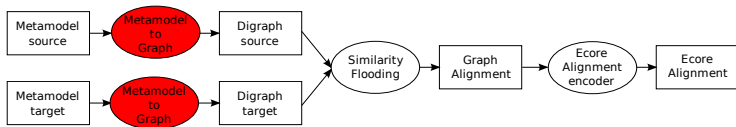
Three steps



The three steps

- 1 From metamodels to graphs
- 2 Application of Similarity Flooding
- 3 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

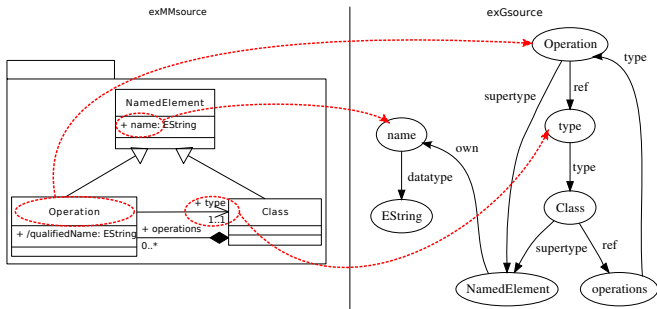
A directed labelled graph representing the model

Objective

- Study the impact on Similarity Flooding of the configuration choice
- Six tested configurations
- Comparison of the results

Configuration *Minimal*

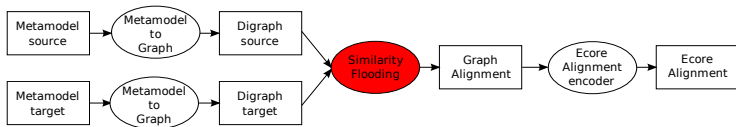
- Metamodel elements are converted into labelled nodes
- Relations are converted into labelled edges
- Derived attributes, references, operations and parameters are ignored



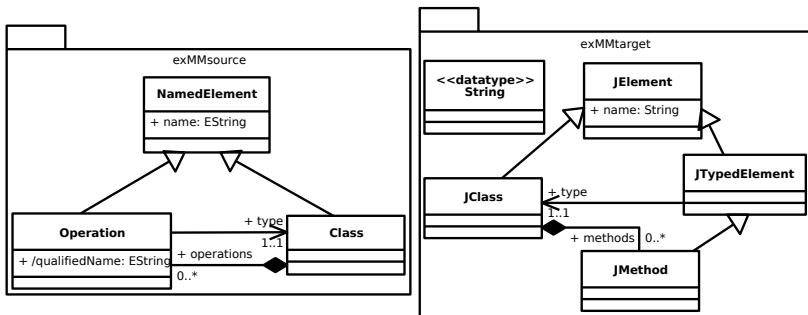
Next configurations

- 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

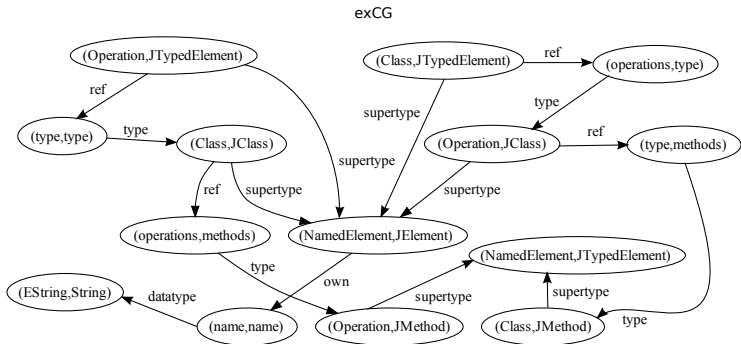
2. Similarity Flooding



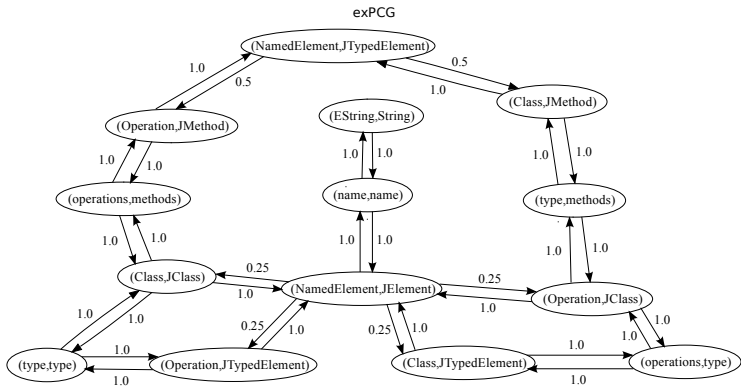
First step : The compatibility graph



First step : The compatibility graph



Second step : propagation graph



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

Compatibility node	Initial similarity value
(NamedElement, JElement)	0.5833334
(name, name)	1.0
(EString, String)	0.85714287
(NamedElement, JTypedElement)	0.6923077
(Operation, JTypedElement)	0.23076922

Principe

- 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 ϵ 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

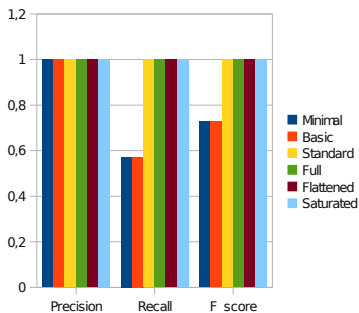
- exMMSource → exMMTarget
- Ecore → Minjava
- Ecore → Kermeta
- Ecore → UML

precision, recall et f_score :

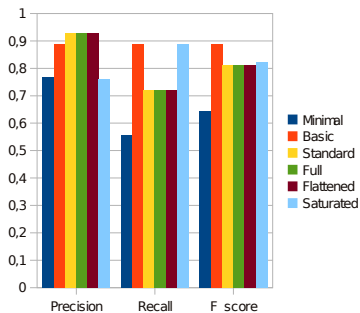
- $precision = \frac{\text{Number_of_Correct_Found_Mappings}}{\text{Number_of_Total_Found_Mappings}}$
- $recall = \frac{\text{Number_of_Correct_Found_Mappings}}{\text{Number_of_Total_Existing_Mappings}}$
- $f_score = \frac{2 \times recall \times precision}{recall + precision}$

Results

exMMSource → exMMTarget

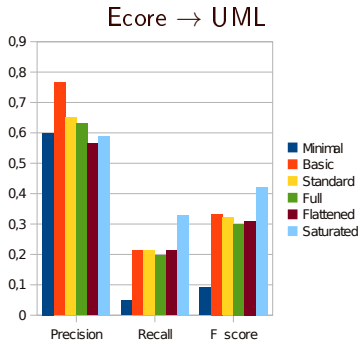
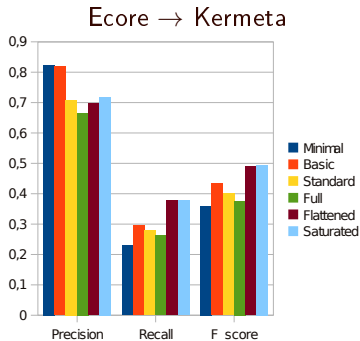


Ecore → Minjava



- 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.

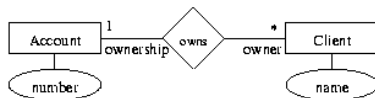
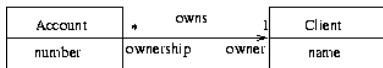
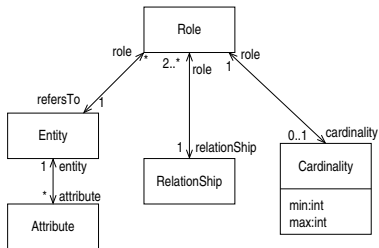
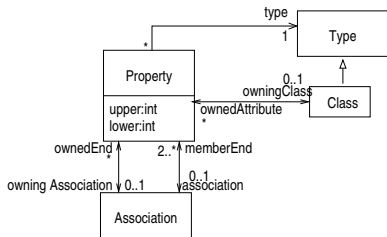
Context

- metamodels : similar to very different ;
- a set of examples

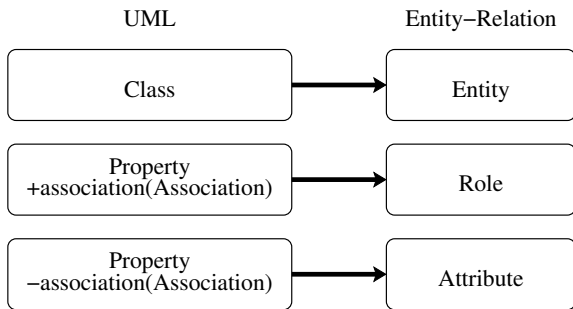
Interest

- use of existing data
- concrete language manipulation
- prior specification of the transformation is not required

Input data

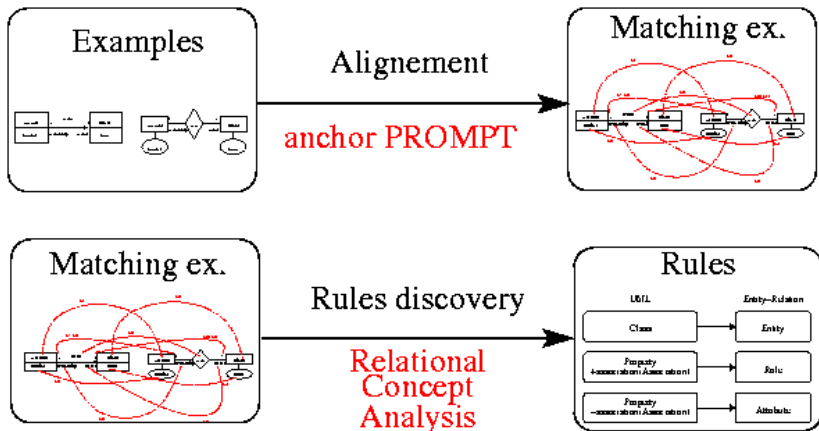


Output data



Transformation rules

Two-step process



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

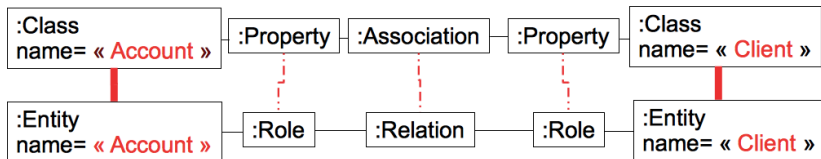
- equality
- substring
- levenshtein (editing) distance

Anchor propagation

Principle

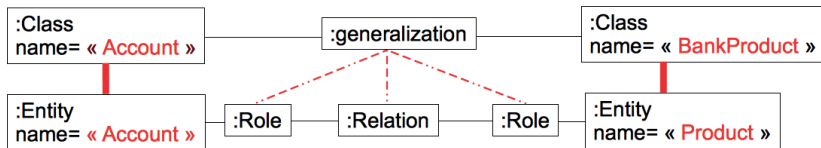
- Inspired by anchorPROMPT approach (noy et al.)
- Align a path in the source model and a path in the target model
- Admit a little size difference between the two paths
- Give weights to matchings, then filter

Anchor-based matching process



Original anchorPROMPT propagation

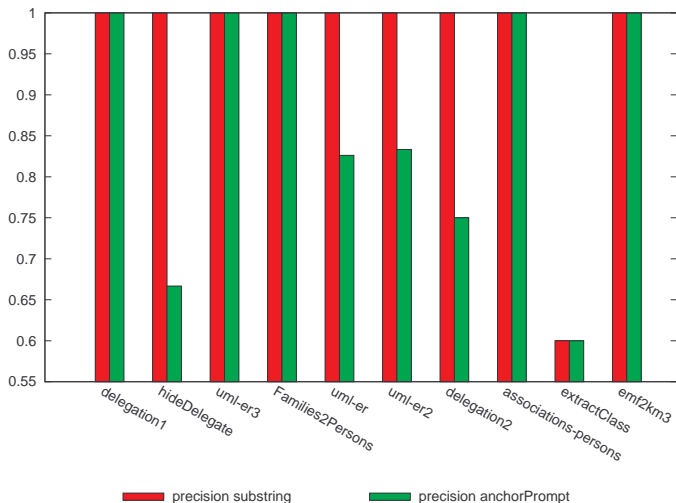
Anchor-based matching process



Extension to paths with different size
e.g. generalization in UML versus is-a relation in ER

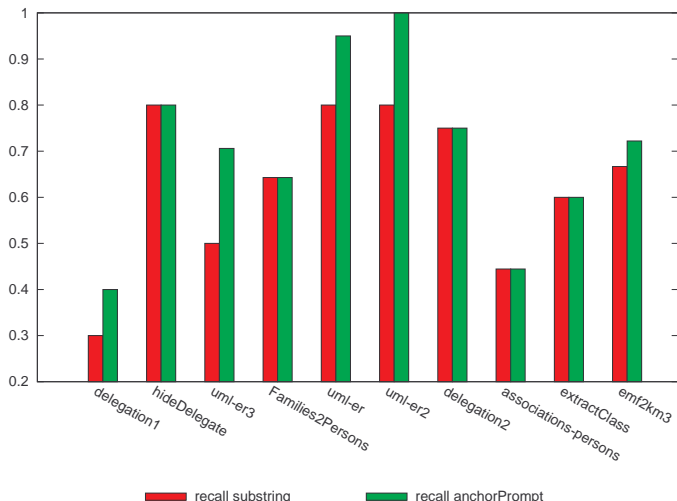
Precision on case study

number of relevant retrieved matches / number of retrieved matches



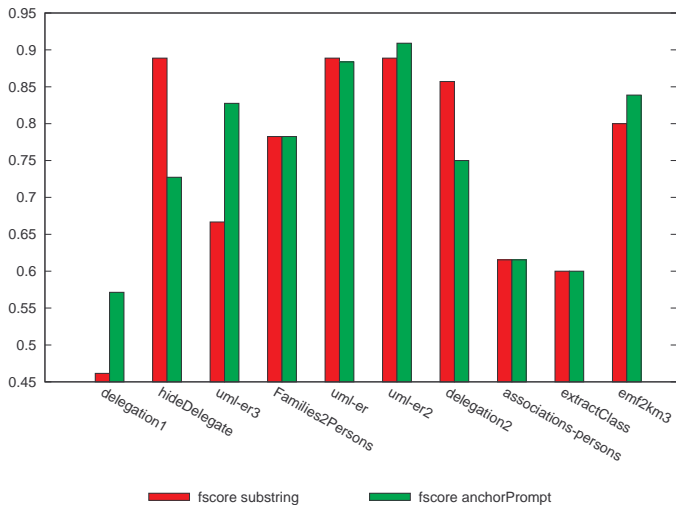
Recall on case study

number of relevant retrieved matches / number of relevant matches



Fscore on case study

$$Fscore = 2 \frac{precision \cdot recall}{precision + recall} \text{ (best is 1)}$$



Rule discovery



Rules discovery

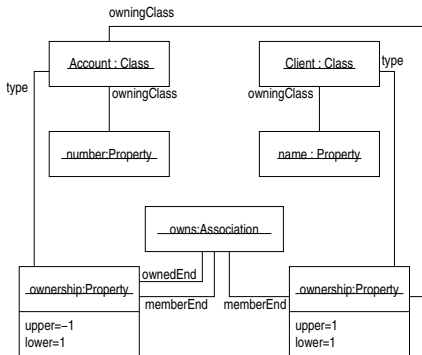
Discovery process' properties

- classification of models elements
- classification of mapping links
- derive rules

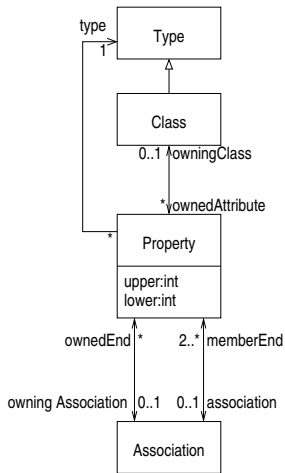
Relational Concept Analysis [Huchard et al. 2007]

- extension of Formal Concept Analysis [Wille1982]
- considers relationships in the classification process

Example data



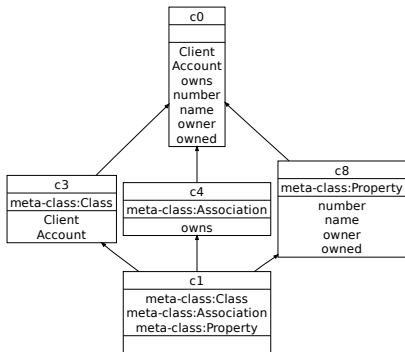
UML model example (seen as instance of the metamodel)



Simplified UML meta-model

Classification of model elements

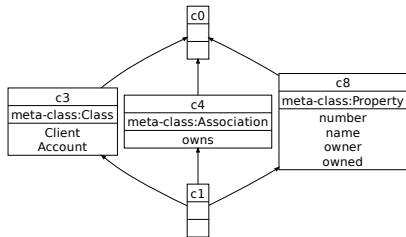
meta-class	Class	Property	Association
Account	X		
Client	X		
owner		X	
owned		X	
owns			X
numero		X	
name		X	



Model element classification using their meta-Classes

Classification of model elements

meta-class	Class	Property	Association
Account	X		
Client	X		
owner		X	
owned		X	
owns			X
numero		X	
name		X	

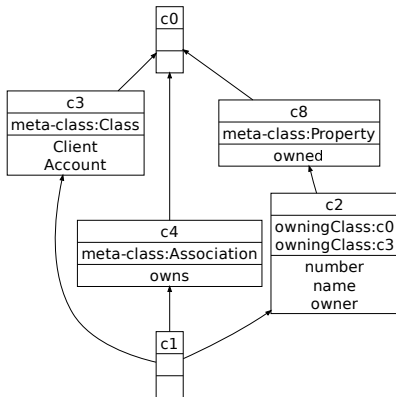


concepts

Model element classification using their meta-Classes

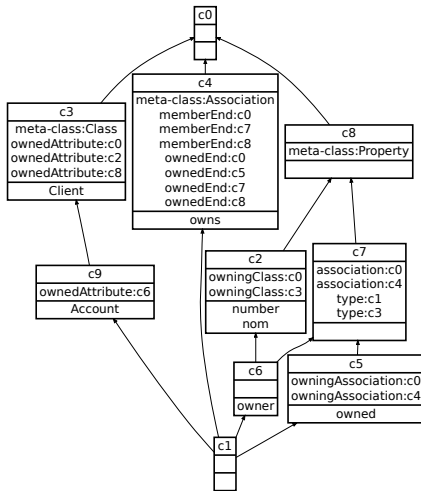
Classification of model elements

owningClass	Account	Client
number	x	
name		x
owner	x	



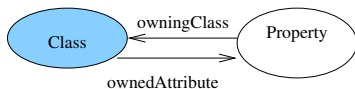
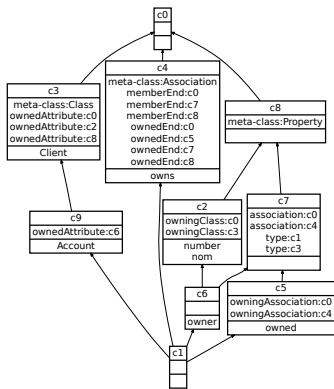
Model elements classification
using their target by the relation *owningClass*.

Model elements classification

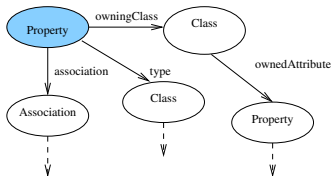


Lattice of the UML model's elements

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

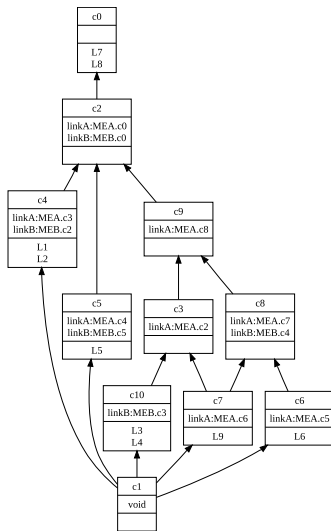
linkA	Account	Client	number	name	owns	owner	owned
L1	x						
L2		x					
L3			x				
L4				x			
L5					x		
L6						x	
L9							x

Table: Relation of mapping links with model source elements

linkB	Account	Client	number	name	owns	owner	owned
L1	x						
L2		x					
L3			x				
L4				x			
L5					x		
L6						x	
L9							x

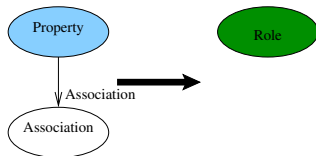
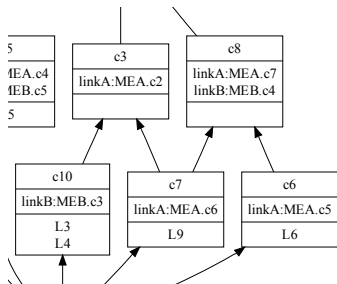
Table: Relation of mapping links with model target elements.

Mappings lattice

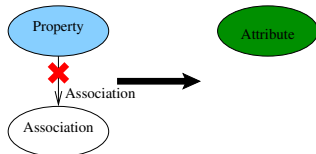


mapping links lattice

Rules extraction



Concept 8 description

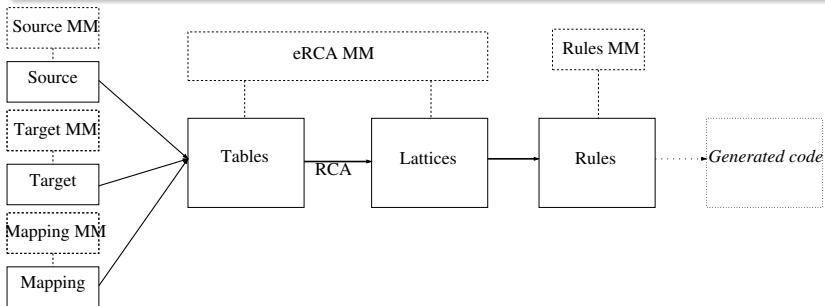


Concept 10 description

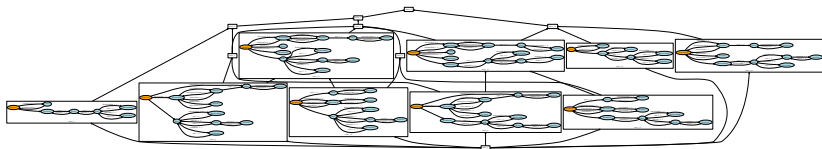
Implementation

Tools

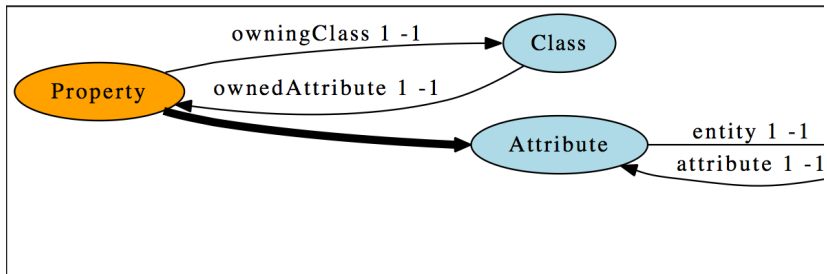
- Eclipse Modeling Framework (EMF)
- lattice generation plugin : eRCA
- template generation tool : Acceleo
- declarative model transformation language : ATL



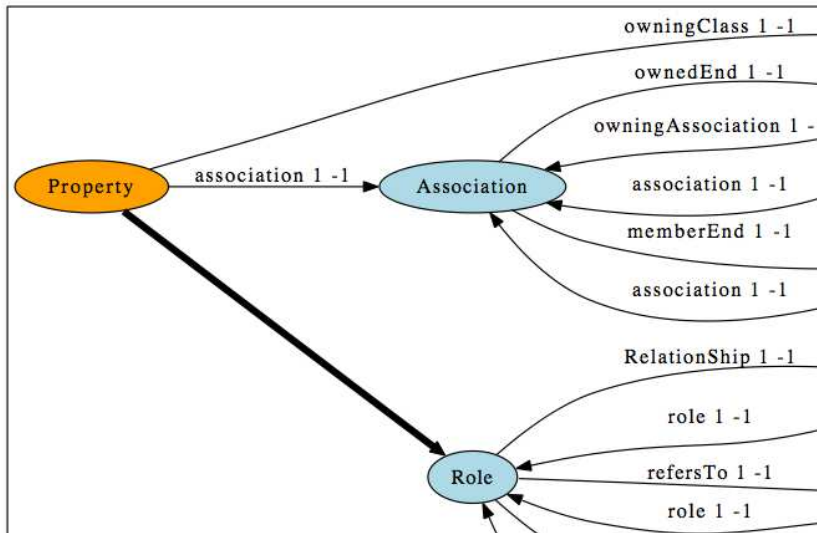
The rule lattice



The rule Property Attribute



The rule Property Role



Validation (in progress)

Table: Data obtained from the case study (ATL zoo)

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

¹ : Family2Person – ² : BibTex2DocBook – ³ : Class2Relation

Outline

- 1 MDE/MT/MTG
- 2 Metamodel alignment based MTG
- 3 Example based MTG
- 4 Towards a global MTG architecture

MM-based MTG

Adapted to similar metamodels

- Ontology-based, pivot ontology (Roser et 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

- J.-R. Falleri, M. Huchard, M. Lafourcade, C. Nebut. Metamodel Matching for Automatic Model Transformation Generation MoDELS UML Conference, 2008, Toulouse, pages 326-340, 2008
- X. Dolques, A. Dogui, J.-R. Falleri, M. Huchard, C. Nebut, F. Pfister. Easing Model Transformation Learning with Automatically Aligned Examples (submitted 2010)
- X. Dolques M. Huchard, C. Nebut. From transformation traces to transformation rules : Assisting model driven engineering approach with formal concept analysis. In Sup. Proceedings of ICCS'09, Moscou, pages 15-29, 2009