Formal and Relational Concept Analysis approaches in Software Engineering:

an overview and an application to learn model transformation patterns in examples

Xavier Dolques, Marianne Huchard, Clémentine Nebut, Hajer Saada

1 INRIA, Centre Inria Rennes - Bretagne Atlantique
2 LIRMM - Université Montpellier 2, CNRS

First ICESE Virtual Workshop on Software Engineering and Artificial Intelligence, may 2011
1 Introduction
2 FCA
3 RCA
4 “Model Transformation By example” approaches
5 Illustrative Example
6 Models Matching
7 Transformation patterns/rules generation
8 Conclusion
Motivations

Software contains plenty of data analysis problems involved in
- the forward engineering process
- the re-engineering tasks
- various analyses

Focusing on Formal Concept Analysis
- an exploratory data analysis / data mining method
- an unsupervised machine learning approach
- produces clusters, classification and implication rules
## Motivations

### Highlight main characteristics of FCA
- defining FCA
- main applications of FCA in SE
- multi-relational data analysis with RCA
- young applications of RCA in SE

### Learning model transformation (MT) patterns
- on examples of MT
- building of MT examples using ontology alignment
- learning MT patterns with RCA
Outline

1. Introduction
2. FCA
3. RCA
4. “Model Transformation By example” approaches
5. Illustrative Example
6. Models Matching
7. Transformation patterns/rules generation
8. Conclusion
### What is FCA?

- A formalization of the philosophical notion of *concept*
- An approach for data analysis and knowledge processing
- Many existing experiences and projects
- Algorithms, graphical representations, tools
- An active research community (3 conf. ICFCA, CLA, ICCS)

*source* [http://people.aifb.kit.edu/jvo/fca4sw/](http://people.aifb.kit.edu/jvo/fca4sw/)
What is a concept?

The concept *bird*

- a set of objects (concept’s extent):
  
- a set of attributes / characteristics (concept’s intent):
  feathers, with a bill, etc.
How concepts are organized?

The concept *flamingo* is a subconcept of the concept *bird*

- inclusion of concept's extents:
  the set of flamingos is included in the set of birds

- inclusion of the concept's intents:
  the attributes of birds are included in the attributes of flamingos
Formal Concept Analysis (FCA)

The formal context

Things that are known about the world

<table>
<thead>
<tr>
<th></th>
<th>flying (fl)</th>
<th>nocturnal (n)</th>
<th>feathered (fe)</th>
<th>migratory (m)</th>
<th>duck-billed (db)</th>
<th>web (w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>flying squirrel</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bat</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ostrich</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>flamingo</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sea-gull</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Concepts can be derived from a formal context

A formal concept is a pair \((X, Y)\) where

- \(Y\) is the set of attributes common to the objects of \(X\)
- \(X\) is the set of objects having all attributes of \(Y\)
The concept lattice: specialization order

more general concepts

more specific concepts
The concept lattice: clusters

- **Flying Animals**
  - Concept_2: nocturnal bat
  - Concept_8: web flying squirrel

- **Birds**
  - Concept_5: feathered ostrich
  - Concept_4: migratory flamingo
  - Concept_7: duck-billed sea-gull

- Concept_0

- Concept_1: flying

- Concept_3

**Flying birds**
The concept lattice: implication rules

- **Concept 0**
  - **Concept 1**
    - flying
  - **Concept 5**
    - feathered
    - ostrich

- **Concept 2**
  - nocturnal
  - bat

- **Concept 6**
  - migratory
  - flamingo

- **Concept 3**

- **Concept 4**

- **Concept 8**
  - web
  - flying squirrel

**Examples:**
- flying and feathered
- migratory

---

**Sections:**
- Introduction
- FCA
- RCA
- By Example
- Example
- Matching
- Transformation Generation
- Conclusion
FCA in Software engineering

A Survey of Formal Concept Analysis Support for Software Engineering Activities, Tilley et al., FCA 2005
... And many research work during the past 5 years

- Requirement Analysis: elaborating requirements [Andelfinger], reconciling stake-holders [Düwel et al.], linking use cases and classes [Böttger et al.]

- Component / Web service classification and retrieval [Lindig, Fisher, Aboud et al., Azmeh et al.]

- Exploring a formal specification [Tilley]

- Dynamic analysis: debug temporal specifications [Ammons et al.], test coverage [Ball], locating features [Eisenbarth et al., Bojic et al.], fault localization [Cellier et al.]
Analysis of legacy systems:
- Configuration structure [Snelting]
- Grouping fields in COBOL systems [Van Deursen et al., Kuipers et al.]
- Migrating COBOL towards Corba components [Canfora]
- Migrating from imperative to OO paradigm [Sahraoui et al., Siff et al., Tonella]
- Reengineering class hierarchies [Snelting et al., Schupp et al., Godin et al., Huchard et al.]
- Detecting patterns [Tonella and Antoniol, Arévalo et al.]
- Order for reading classes [Dekel]
- Bad smell correction [Bhatti et al.]
- Conceptual code exploration [Cole et al.]
- Aspect mining [Tonella and Ceccato, Tourwé and Mens]
- Access-guided client class extraction for Eiffel [Ardourel and Huchard]
- Mining Source Code for Structural Regularities [Lozano et al.]
Outline

1. Introduction
2. FCA
3. RCA
4. “Model Transformation By example” approaches
5. Illustrative Example
6. Models Matching
7. Transformation patterns/rules generation
8. Conclusion
Relational Concept Analysis (RCA)

- Extend the purpose of FCA for taking into account relations between objects
- The RCA process relies on the following main points:
  - A relational model based on the entity-relationship model
  - A conceptual scaling process allowing to represent relations between objects as relational attributes
  - An iterative process for designing a concept lattice where concept intents include non-relational and relational attributes.
- RCA provides relational structures that can be represented as ontology concepts within a knowledge representation formalism such as description logics (DLs).

A relational model based on the entity-relationship model ...

Pizza story
Relational Concept Analysis (RCA)

Pizza story

Pizza data

- four object/attribute contexts
  - $K_{People} \subseteq \text{People} \times \text{people names}$
  - $K_{Pizza} \subseteq \text{Pizza} \times \text{pizza names}$
  - $K_{Food} \subseteq \text{Food item} \times \text{food names}$
  - $K_{Country} \subseteq \text{Country} \times \text{country names}$

- four object/object contexts
  - eats $\subseteq \text{People} \times \text{Pizza}$
  - contains $\subseteq \text{Pizzas} \times \text{Food item}$
  - producedIn $\subseteq \text{Food item} \times \text{Country}$
  - hasForNational $\subseteq \text{Country} \times \text{People}$
### Formal contexts

<table>
<thead>
<tr>
<th>$K_{People}$</th>
<th>Amedeo</th>
<th>Flavia</th>
<th>Dagrun</th>
<th>Lars</th>
<th>Uma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amedeo</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flavia</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dagrun</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lars</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Uma</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$K_{Pizza}$</th>
<th>Dronning</th>
<th>Kampanje</th>
<th>Margherita</th>
<th>Marina</th>
<th>Norwegian</th>
<th>Regina</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dronning</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kampanje</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Margherita</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marina</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Norwegian</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Regina</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$K_{Ingredients}$</th>
<th>basilic</th>
<th>chicken</th>
<th>cream</th>
<th>ham</th>
<th>mozzarella</th>
<th>mushroom</th>
<th>olive</th>
<th>tomato</th>
<th>salmon</th>
<th>swisscheese</th>
</tr>
</thead>
<tbody>
<tr>
<td>basilic</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chicken</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cream</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ham</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mozzarella</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mushroom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>olive</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>tomato</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>salmon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>swisscheese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$K_{Country}$</th>
<th>Italy</th>
<th>Norway</th>
<th>Switzerland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Italy</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Norway</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Switzerland</td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>
Relational context: $R_{eats}$

<table>
<thead>
<tr>
<th></th>
<th>Dronning</th>
<th>Kampanje</th>
<th>Margherita</th>
<th>Marina</th>
<th>Norwegian</th>
<th>Regina</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amedeo</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>Flavia</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dagrun</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Lars</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>Uma</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>
### Relational context: $R_{\text{contains}}$

<table>
<thead>
<tr>
<th></th>
<th>basilic</th>
<th>chicken</th>
<th>cream</th>
<th>ham</th>
<th>mozzarella</th>
<th>mushroom</th>
<th>olive</th>
<th>tomato</th>
<th>salmon</th>
<th>swisscheese</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dronning</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kampanje</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Margherita</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marina</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Norwegian</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regina</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Relational context: $R_{producedIn}$

<table>
<thead>
<tr>
<th></th>
<th>Italy</th>
<th>Norway</th>
<th>Switzerland</th>
</tr>
</thead>
<tbody>
<tr>
<td>basilic</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chicken</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>cream</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>ham</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mozzarella</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mushroom</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>olive</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tomato</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>salmon</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>swisscheese</td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>
### Relational context: $R_{\text{hasForNational}}$

<table>
<thead>
<tr>
<th></th>
<th>Amedeo</th>
<th>Flavia</th>
<th>Dagrun</th>
<th>Lars</th>
<th>Uma</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Italy</strong></td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Norway</strong></td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td><strong>Switzerland</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Relational Context Family (RCF)

A RCF $\mathcal{F}$ is a pair $(K, R)$ with:

- $K$ is a set of formal contexts $K_i = (O_i, A_i, I_i)$
- $R$ is a set of relational contexts $R_j = (O_k, O_l, I_j)$,

Pizza RCF

$K = K_{\text{People}}, K_{\text{Pizza}}, K_{\text{Food}}, K_{\text{Country}}$

$R = R_{\text{eats}}, R_{\text{contains}}, R_{\text{producedIn}}, R_{\text{hasForNational}}$
An iterative approach (RCA)

Learned concepts are used in a next step to learn more
RCA - Step 0 - Initial Lattices

Introduction  FCA  RCA  By Example  Example  Matching  Transformation Generation  Conclusion
Integrating concepts in the relational contexts

Amedeo eats Margherita ; Margherita ∈ extent(Concept_11)
→ ∃p ∈ Concept_11, s.t. Amedeo eats p
→ (Amedeo, eats:Concept_11)
→ (Amedeo, Concept_11) belongs to the existentially scaled relation eat*, (Amedeo, ∃ eat :Concept_11) stands

<table>
<thead>
<tr>
<th></th>
<th>Amedeo</th>
<th>Flavia</th>
<th>Dagrun</th>
<th>Lars</th>
<th>Uma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amedeo</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flavia</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dagrun</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lars</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Uma</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>eats : Concept_7</th>
<th>eats : Concept_8</th>
<th>eats : Concept_11</th>
<th>eats : Concept_12</th>
<th>eats : Concept_13</th>
<th>eats : Concept_14</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
RCA - Lattices at step 1

Concept 0
- eats : Concept 7

Concept 34
- eats : Concept 14

Concept 32
- eats : Concept 12

Concept 33
- eats : Concept 13

Concept 1
- Amedeo
  - eats : Concept 11
  - Amedeo

Concept 4
- Daqrun
  - eats : Concept 14

Concept 5
- Flavia
  - Lars
  - Flavia

Concept 6
- Uma
  - eats : Concept 8
  - Uma

Concept 2
- People

Concept 7
- contains : Concept 15

Concept 39
- contains : Concept 24

Concept 35
- contains : Concept 19
  - contains : Concept 28

Concept 36
- contains : Concept 20
  - contains : Concept 22

Concept 37
- contains : Concept 21

Concept 38
- contains : Concept 23

Concept 10
- Kampanje
  - contains : Concept 18
  - Kampanje

Concept 13
- Norwegian
  - contains : Concept 25

Concept 8
- Dronning
  - Dronning

Concept 14
- Regina
  - Regina

Concept 11
- Margherita
  - contains : Concept 16
  - Margherita

Concept 12
- Marina
  - Marina

Concept 15
- producedIn : Concept 27

Concept 40
- producedIn : Concept 29

Concept 42
- producedIn : Concept 31

Concept 41
- producedIn : Concept 30

Concept 27

Concept 43
- hasForNational : Concept 31
  - Swiss
  - Swiss

Concept 30
- hasForNational : Concept 43
  - hasForNational : Concept 4
  - Italian

Concept 28
- hasForNational : Concept 30
  - hasForNational : Concept 6
  - Norway

Concept 29

Concept 21
- mozzarella
  - olive

Concept 23
- olive
tomato
basil

Concept 24
- olive
tomato
basil

Concept 16
- ham

Concept 20
- ham

Concept 26
- swisscheese
swisscheese

Concept 19
- cream
  - salmon
  - chicken

Concept 25
- salmon
  - chicken

Concept 22
- mushroom

Concept 17
- Food

Country
- Norway
- Italian
- Italy

Introduction
FCA
RCA
By Example
Example
Matching
Transformation Generation
Conclusion
RCA - Lattices at step 2
RCA - Lattices at step 3

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion
An excerpt of the iteration
Applications

- **UML class diagram refactoring**

- **UML Use case diagram refactoring**
  * X. Dolques, M. Huchard, C. Nebut, and P. Reitz. Fixing generalization defects in UML use case diagrams. CLA 2010: 247-258

- **Blob design defect correction**

- **Extracting architectures in object-oriented software**
Applications

- **Learning model Transformation patterns in MDE**

- **Classification of web services**

- **Ontology construction**

- **Ontology pattern extraction**

- **Ontology restructuring**
A synthesis on RCA

- an iterative method to produce abstractions
- variations on scaling operators: \( \exists, \forall, \forall \exists, \geq n r : c \), etc. (on relational contexts and on steps)
- object-attribute concept posets can be built instead of lattices to limit the complexity
- opportunities for enhancing application of FCA to software engineering domain

Tools

- Galicia: http://galicia.sourceforge.net/
- eRCA: http://code.google.com/p/erca/
1. Introduction
2. FCA
3. RCA
4. “Model Transformation By example” approaches
5. Illustrative Example
6. Models Matching
7. Transformation patterns/rules generation
8. Conclusion
Context and Motivations

Context
- Model driven development
- Development of a model transformation
- Source and target metamodels require domain experts

Motivations
- Ease and speed up the development process of model transformations
- Improve the integration of domain expert in the process
The “By Example” approach

Informal Example
The “By Example” approach
The “By Example” approach

Informal Example

- Model 1
  - Metamodel 1
    - Conforming to Metamodel 1

- Model 2
  - Metamodel 2
    - Conforming to Metamodel 2

*modelised by*  
*conforming to*
The “By Example” approach

Informal Example

- Model 1
- Model 2

Metamodel 1
- conforming to
- modelised by
- transformation

- Model 2
- conforming to

Metamodel 2

- transformation
The “By Example” approach

UML

Person

name

Class

ownedAttribute

Property

conforming to

Entity-Relationship

Person

name

Entity

Attribute

attribute

conforming to

instance (Class, x) ∧ ownedAttribute (x) ≠ ∅ ⇒ instance (Entity, t(x)) ∧ (y ∈ ownedAttribute (x) → t(y) ∈ attribute (t(x)))

instance (Property, x) ∧ association (x) = ∅ ⇒ instance (Attribute, t(x))
The “By Example” approach

UML
Person
name
Class
Property
ownedAttribute
Entity
Attribute
conforming to
(1)
(2)

Entity-Relationship
Person
name
Entity
Attribute
conforming to

instance (Class, x) \land ownedAttribute (x) \neq \emptyset \Rightarrow instance (Entity, t(x)) \land (y \in ownedAttribute (x) \rightarrow t(y) \in attribute (t(x)))

instance Property, x \land association (x) = \emptyset \Rightarrow instance Attribute, t(x)
The “By Example” approach

(1) $instance(Class, x) \land ownedAttribute(x) \neq \emptyset \Rightarrow instance(Entity, t(x)) \land (y \in ownedAttribute(x) \rightarrow t(y) \in attribute(t(x)))$

(2) $instance(Property, x) \land association(x) = \emptyset \Rightarrow instance(Attribute, t(x))$
“By Example” approaches

[Balogh et Varró(2009)]

- **Input matching**: set of typed couples of elements
- **Matching creation**: manually
- **Input specific development**: none
- **Learning principle**: Inductive Logic Programming
- **Output data**: transformation rules (VIATRA)
“By Example” approaches

[Wimmer et al. (2007)]

- **Input matching**: set of couples of elements
- **Matching creation**: manually
- **Input specific development**: explicit constraints of the transformation from concrete to abstract syntax
- **Learning principle**: ad hoc method
- **Output data**: ATL code
“By Example” approaches

[Kessentini et al. (2008)]

- **Input matching**: set of block couples
- **Matching creation**: manually
- **Input specific development**: none
- **Learning principle**: metaheuristics
- **Output data**: a transformed model
“By Example” approaches

Our approach

- **Input matching**: set of couples of elements
- **Matching creation**: matching assisted by tool
- **Input specific development**: none
- **Learning principle**: Relational Concept Analysis
- **Output data**: specification of transformation rules ordered in a lattice
General approach

- General approach conforms to input/output examples models.
- Transformation generation.

Icons: http://www.openclipart.org (Public Domain)

Examples Models
Matching Transformation

Model 1
Model 2
Metamodel 1
Metamodel 2

Examples
Models
Matching
Transformation
rules
1. Introduction

2. FCA

3. RCA

4. “Model Transformation By example” approaches

5. Illustrative Example

6. Models Matching

7. Transformation patterns/rules generation

8. Conclusion
Illustrative Example

Icons: http://www.openclipart.org (Public Domain)

- Source MetaModel
- Target MetaModel

Examples

- Source Model
- Target Model

Model 1

- Modeled by

Model 2

- Modeled by

Metamodel 1

- Conforms to

Metamodel 2

- Conforms to

Transformation rules

Matching Generation

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion
Tranformation Example

UML model

conforming to

Entity-Relationship model

conforming to

UML metamodel

conforming to

Entity-Relationship metamodel
Metamodels involved in the transformation
Metamodels involved in the transformation
Excerpt of UML metamodel
Metamodels involved in the transformation

Entity-Relationship metamodell
Models involved in the transformation

UML model

conforming to

Entity-Relationship model

conforming to

UML metamodel

conforming to

Entity-Relationship metamodel
Models of our Example
A UML Model in concrete syntax

![UML Diagram]

- **Author**
  - firstName
  - lastName
  - has a *year* dimension
  - writes foreword for *author* dimension
  - 0..1 *forewordAuthor* dimension

- **Text**
  - title
  - has a *text* dimension
  - 1..* *work* dimension
  - *writes foreword for* *forewordWritten* dimension

- **Style**
  - has a *style* dimension

- **Novel**

- **Poetry**
Models of our Example
A UML Model in abstract syntax
Models of our Example
A UML model (excerpt)
Models of our Example
An Entity-Relationship model in concrete syntax

Introduction  FCA  RCA  By Example  Example  Matching  Transformation Generation  Conclusion
Models of our Example
An Entity-Relationship model in concrete syntax
Models of our Example
An Entity-Relationship model (excerpt)
Outline

1. Introduction
2. FCA
3. RCA
4. “Model Transformation By example” approaches
5. Illustrative Example
6. Models Matching
7. Transformation patterns/rules generation
8. Conclusion
Models Matching

Icons: http://www.openclipart.org (Public Domain)
Hypotheses

- the informal starting example contains named elements: those elements are to be found in the two models
- source and target models structure are close enough
State of the Art

Constraints of the matching methods that can be applied

- can be applied on a graph structure (not only a tree)
- not strictly based on semantic analysis

Relevant matching approaches

- Similarity Flooding [Melnik et al., 2002]
- OLA [Euzenat et al., 2004]
- Anchor Prompt [Noy et Musen, 2001]

Advantage of anchorPrompt

does not use similarity on relations names (relations names come from the metamodel level)
Description of process

Matching using Attributes values
- enumeration of values in the models
- matching of similar values
- matching of the elements containing those values

Matching Propagation
- using structure similarity assumption
Attribute Instances
In the UML source model
Attribute Instances
In the UML source model (excerpt)
Attribut Instances

In the Entity-Relationship target model
Attribute Instances

In the Entity-Relationship target model (excerpt)
Value comparison

A source attribute instance and a target attribute instance match if all the following conditions are respected:

- they have the same value
- this value appears only once in the source model and the target model
Value matchings in attributes
A first Model Matching

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion
Propagation of the machining
The AnchorPrompt approach Adaptation

\[ W(x, y) = 1 - \left| \frac{\text{index}(x)}{\text{length}(X) - 1} - \frac{\text{index}(y)}{\text{length}(Y) - 1} \right| \]
Introduction  FCA  RCA  By Example  Example  Matching  Transformation Generation  Conclusion
Similarities propagation
Empirical results

Precision calculation

- Precision substring
- Precision anchor Prompt

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion
To address the model matching problem

- **Starting with 2 assumptions**
  - the starting example contains named element
  - the models to match are structurally close

- **We propose a process generating a model matching**
  - using attributes values for a first matching with high confidence level
  - using an adaptation of AnchorPROMPT for extending the first matching
Outline

1. Introduction
2. FCA
3. RCA
4. “Model Transformation By example” approaches
5. Illustrative Example
6. Models Matching
7. Transformation patterns/rules generation
8. Conclusion
Transformation rules generation
Input Data

UML model
conforming to

Entity-Relationship model
conforming to

UML metamodel

Entity-Relationship metamodel

Introduction  FCA  RCA  By Example  Example  Matching  Transformation  Generation  Conclusion
Input Data (excerpt)
Approach

Consider the properties of the model elements:
- their class
- their relations with their neighbors
- the properties of their neighbors

Classify the different properties from the examples
Classify the matching links considering the classification of the properties of their extremities

Formal Concept Analysis allows the classification of a set of objects considering their attributes
Transformation of models in tables

- Metamodel elements contexts
  - Source metamodel context
  - Target metamodel context
- Model elements contexts
  - Source model context
  - Target model context
- Matching links context
- Relations
  - between model source elements and their class from the source metamodel
  - between model target elements and their class from the target metamodel
  - between model source elements: e.g. ownedAttribute
  - between model target elements: e.g. attribute
  - between matching links and their source from the source model
  - between matching links and their target from the target model

The management of relations requires RCA to iterate
Lattice of model elements
Source model (UML)

Concept 49

Property

Class

type
Lattice of model element
Target model (Entity-Relationship)
rules lattice (excerpt)
Rules examples

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion
Hypothesis
- closed world
- needs a good cover of the transformation

Different kinds of characteristics
- needed characteristics
- allowed characteristics
- forbidden characteristics
Needed characteristics

---

Introduction  FCA  RCA  By Example  Example  Matching  Transformation Generation  Conclusion
Allowed characteristics

rules_11 - Concept_81
Forbidden characteristics

rules_11 - Concept_81

[Diagram of class and entity relationships with properties and attributes]

Concept_42 (S: 1)
- metaClassA : Property
  - elementA_2
  - elementA_3
  - elementA_5
  - elementA_14

Concept_49 (S: 1)
- type : Concept_41
  - elementA_16

Type: [0..0]
Entity: [0..*]
Empirical study

<table>
<thead>
<tr>
<th>name</th>
<th>size of MapLinks lattice</th>
<th>number of extracted rules</th>
<th>premise size</th>
<th>premise depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>associations-persons</td>
<td>9</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Book2Publication</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>delegation1</td>
<td>47</td>
<td>12</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>delegation2</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>Disaggregation</td>
<td>22</td>
<td>10</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>Ecore2Class</td>
<td>14</td>
<td>8</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>EliminateRedundantInheritance</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>emf2km3</td>
<td>26</td>
<td>19</td>
<td>14</td>
<td>52</td>
</tr>
<tr>
<td>EquivalenceAttributesAssociations</td>
<td>47</td>
<td>18</td>
<td>11</td>
<td>36</td>
</tr>
<tr>
<td>extractClass</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Families2Persons_final</td>
<td>11</td>
<td>8</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>hideDelegate</td>
<td>53</td>
<td>22</td>
<td>17</td>
<td>117</td>
</tr>
<tr>
<td>IntroducePrimaryKey</td>
<td>31</td>
<td>17</td>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td>IntroducingInterface</td>
<td>38</td>
<td>11</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>JavaSource2Table</td>
<td>17</td>
<td>10</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>KM32EMF</td>
<td>18</td>
<td>15</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>KM32Problem</td>
<td>37</td>
<td>28</td>
<td>28</td>
<td>378</td>
</tr>
<tr>
<td>uml-er</td>
<td>35</td>
<td>24</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>uml-er-iccs</td>
<td>21</td>
<td>12</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>uml-er2</td>
<td>25</td>
<td>15</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>uml-er3</td>
<td>40</td>
<td>19</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>UML2ClassDiagram_to_KM3</td>
<td>200</td>
<td>40</td>
<td>28</td>
<td>187</td>
</tr>
</tbody>
</table>
## Empirical study

<table>
<thead>
<tr>
<th>nom</th>
<th>conclusion size</th>
<th></th>
<th>conclusion depth</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
<td>avg.</td>
<td>std. dev.</td>
</tr>
<tr>
<td>associations-persons</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0.91</td>
</tr>
<tr>
<td>Book2Publication</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>delegation1</td>
<td>7</td>
<td>17</td>
<td>7</td>
<td>2.89</td>
</tr>
<tr>
<td>delegation2</td>
<td>15</td>
<td>26</td>
<td>20</td>
<td>4.96</td>
</tr>
<tr>
<td>Disaggregation</td>
<td>7</td>
<td>11</td>
<td>7</td>
<td>1.26</td>
</tr>
<tr>
<td>Ecore2Class</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>2.26</td>
</tr>
<tr>
<td>EliminateRedundantInheritance</td>
<td>7</td>
<td>11</td>
<td>8</td>
<td>1.63</td>
</tr>
<tr>
<td>emf2km3</td>
<td>11</td>
<td>20</td>
<td>14</td>
<td>3.59</td>
</tr>
<tr>
<td>EquivalenceAttributesAssociations</td>
<td>9</td>
<td>38</td>
<td>14</td>
<td>8.37</td>
</tr>
<tr>
<td>extractClass</td>
<td>16</td>
<td>20</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>Families2Persons_final</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0.79</td>
</tr>
<tr>
<td>hideDelegate</td>
<td>17</td>
<td>186</td>
<td>39</td>
<td>37.75</td>
</tr>
<tr>
<td>IntroducePrimaryKey</td>
<td>9</td>
<td>14</td>
<td>9</td>
<td>1.21</td>
</tr>
<tr>
<td>IntroducingInterface</td>
<td>2</td>
<td>20</td>
<td>9</td>
<td>4.53</td>
</tr>
<tr>
<td>JavaSource2Table</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0.63</td>
</tr>
<tr>
<td>KM32EMF</td>
<td>14</td>
<td>52</td>
<td>21</td>
<td>10.23</td>
</tr>
<tr>
<td>KM32Problem</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>uml-er</td>
<td>2</td>
<td>15</td>
<td>9</td>
<td>3.61</td>
</tr>
<tr>
<td>uml-er-iccs</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>0.29</td>
</tr>
<tr>
<td>uml-er2</td>
<td>2</td>
<td>15</td>
<td>9</td>
<td>3.71</td>
</tr>
<tr>
<td>uml-er3</td>
<td>2</td>
<td>15</td>
<td>9</td>
<td>3.47</td>
</tr>
<tr>
<td>UML2ClassDiagram_to_KM3</td>
<td>18</td>
<td>45</td>
<td>22</td>
<td>6.06</td>
</tr>
</tbody>
</table>
The generation of transformation patterns is the result of the following steps:

- The transformation of the examples into relational contexts
- The application of the process RCA
- The interpretation of the final lattices
Outline

1. Introduction
2. FCA
3. RCA
4. “Model Transformation By example” approaches
5. Illustrative Example
6. Models Matching
7. Transformation patterns/rules generation
8. Conclusion
An approach with tool for assisting the development of model transformations

Icons: http://www.openclipart.org (Public Domain)

Conclusion

Introduction  FCA  RCA  By Example  Example  Matching  Transformation Generation  Conclusion
Contribution

- Implementation independent with transformation metamodels
- Model matching assisted
- Result usable thanks to lattices
- Result genericity
Model matching
- Improving precision on the AnchorPROMPT adaptation
- Adapt other matching approaches for model matching

Transformation rules generation
- Building relations in final model for not simple case
- Improving interaction with user
- Insisting on approach validation

Evolutions
- Adapt the approach to other problems
- Investigating complementarity with other approaches
Thank you for your attention
Metamodel matching

Lopes et al. (2005)
Generating transformation definition from mapping specification: Application to web service platform.
*In CAiSE’05, LNCS 3520, pages 309–325, 2005.*

Del Fabro et Valduriez (2007)
Semi-automatic model integration using matching transformation and weaving models.

Falleri et al. (2008)
Meta-model Matching for Automatic Model Transformation Generation.
Balogh et Varró (2009)
Model transformation by example using inductive logic programming.


Kessentini et al. (2008)
Model Transformation as an Optimization Problem.

Wimmer et al. (2007)
Towards model transformation generation by-example.