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

To cite this version:

HAL Id: lirmm-00616272
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00616272v2
Submitted on 2 Dec 2011

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.
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
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
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
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 (S)</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bat (B)</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ostrich (O)</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>flamingo (F)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sea-gull (G)</td>
<td>x</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

Concept_1
flying

Concept_3

Concept_2
nocturnal
bat

Concept_5
feathered
ostrich

Concept_4

Concept_6
migratory
flamingo

Concept_8
web
flying squirrel

Concept_7
duck-billed
sea-gull

more specific concepts
The concept lattice: clusters

Flying Animals

Birds

Flying birds
The concept lattice: implication rules

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion

Concept_0

Concept_1
- flying

Concept_2
- nocturnal
- bat

Concept_3

Concept_4

Concept_5
- feathered
- ostrich

Concept_6
- migratory
- flamingo

Concept_7
- duck-billed
- sea-gull

Concept_8
- web
- flying squirrel

flying and feathered

migratory
... 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

![Diagram of entities and relationships: People -> Pizza, People -> Country, Country -> Food, People -> Food]

- People
  - Name
  - HasForNational
- Pizza
  - Name
- Country
  - Name
  - ProducedIn
- Food
  - Name
  - Contains
Relational Concept Analysis (RCA)

Pizza story

Pizza data

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

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

#### $K_{\text{People}}$

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

#### $K_{\text{Pizza}}$

<table>
<thead>
<tr>
<th>Dronning</th>
<th>Kampanje</th>
<th>Margherita</th>
<th>Marina</th>
<th>Norwegian</th>
<th>Regina</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### $K_{\text{Ingredients}}$

<table>
<thead>
<tr>
<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>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### $K_{\text{Country}}$

<table>
<thead>
<tr>
<th>Italy</th>
<th>Norway</th>
<th>Switzerland</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

---

**Introduction**

FCA

RCA

By Example

Example

Matching

Transformation Generation

Conclusion
### 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></td>
<td></td>
<td>×</td>
<td></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>
<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>
<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></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>

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion
### 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>Italy</td>
<td>( \times )</td>
<td>( \times )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Norway</td>
<td></td>
<td></td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>Switzerland</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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**

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

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

Learned concepts are used in a next step to learn more
Scaling relations

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
RCA - Lattices at step 1

Introduction
FCA
RCA
By Example
Example
Matching
Transformation Generation
Conclusion
RCA - Lattices at step 2
RCA - Lattices at step 3
An excerpt of the iteration

STEP 1
- Concept 40
  - basilic
  - ham
  - mozza
  - olive
  - tomato
  - chicken
  - mushroom
  - salmon
  - cream
  - swiss cheese

STEP 2
- Concept 46
  - Dronning
  - Marghrita

STEP 3
- Concept 49
  - Flavia
  - Daqrun
  - Amedeo
  - Uma

- Lars
- Norwegian

Produced In

- Italy

Concept 28

STEP 4
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/
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

Informal Example

modelised by

Model 1

modelised by

Model 2

conforming to

Metamodel 1

conforming to

Métamodèle 2
The “By Example” approach

Informal Example

modelised by

Model 1

conforming to

Metamodel 1

modelised by

Model 2

conforming to

Métamodèle 2
The “By Example” approach

Informal Example

modelised by

Model 1

modelised by

Model 2

conforming to

Metamodel 1

transformation

conforming to

Métamodèle 2
The “By Example” approach
The “By Example” approach
The “By Example” approach

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

(2) \( \text{instance}(\text{Property}, x) \land \text{association}(x) = \emptyset \Rightarrow \text{instance}(\text{Attribute}, t(x)) \)
“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

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

**Examples**

- Source MetaModel
- Target MetaModel

**Models**

- Source Model
- Target Model

**Matching**

- MANDARINE
- BERCAMOTE

**Transformation generation**

- Transformation rules

**Conclusion**
Illustrative Example

- Model 1 conforms to Metamodel 1
- Model 2 conforms to Metamodel 2

Example Matching Transformation Generation

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

UML model

conforming to

Entity-Relationship model

conforming to

UML metamodel

conforming to

Entity-Relationship metamodel
Metamodels involved in the transformation

UML model

conforming to

UML metamodel

Entity-Relationship model

conforming to

Entity-Relationship metamodel

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion
Metamodels involved in the transformation
Excerpt of UML metamodel
Metamodels involved in the transformation
Entity-Relationship metamodel
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
Models of our Example
A UML Model in abstract syntax

Author: Class
  name="Author"
  - lastName: Property
     name="lastName"
     lowerBound=1
     upperBound=1
  - firstName: Property
     name="firstName"
     lowerBound=1
     upperBound=1

writes: AssociationClass
  name="Writes"

year: Property
  name="year"
  lowerBound=1
  upperBound=1
work: Property
  name="work"
  lowerBound=0
  upperBound=-1

forewordAuthor: Property
  name="forewordAuthor"
  lowerBound=0
  upperBound=1

forewordWritten: Property
  name="forewordWritten"
  lowerBound=0
  upperBound=1

work: Property
  name="work"
  lowerBound=0
  upperBound=1

style: Property
  name="work"
  lowerBound=1
  upperBound=1
text: Property
  name="text"
  lowerBound=0
  upperBound=-1

Novel: Class
  name="Novel"
  - title: Property
     name="title"
     lowerBound=1
     upperBound=1
  - text: Property
     name="text"
     lowerBound=0
     upperBound=-1
g1: Generalization
  name="g1"
g2: Generalization
  name="g2"
Poesy: Class
  name="Poetry"
  - style: Property
     name="work"
     lowerBound=1
     upperBound=1

has a: Association
  name="has a"

writes foreword for: Association
  name="writes foreword for"
Models of our Example
A UML model (excerpt)
Models of our Example
An Entity-Relationship model in concrete syntax

Author
firstName lastName
writes foreword for
writes Text
year
Style
Novel Poetry
title
(1,N) (1,N) (1,1) (1,1) (0,1) (0,1) (1,1) (0,N)
work
author
is a is a
has a
58

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)

Text:Entity
name="Text"
has a:Relationship
name="has a"
Style:Entity
name="Style"

Text

(1,1) has a (0,N) Style

Title:Attribute
name="title"

Text:Role
name="text"

Style:Role
name="work"

textC:Cardinality
min=0
max=-1

styleC:Cardinality
min=1
max=1

Title

(1,1) has a (0,N) Style

Text:Entity
name="Text"

Style:Entity
name="Style"
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)
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
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
In the UML source model (excerpt)

```
Text
  title

Style

has a

style

Text
  text

Style
  name="Style"
  title:Property
    name="title"
    lowerBound=1
    upperBound=1

has a:Association
  name="has a"

text:Property
  name="text"
  lowerBound=0
  upperBound=-1

style:Property
  name="work"
  lowerBound=1
  upperBound=1
```

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion
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

Text: Entity
name="Style"
Text: Entity
name="Text"
has a: Relationship
name="has a"

1 *
style text

Title: Property
name="title"
lowerBound=1
upperBound=1

Text: Class
name="Text"

Style: Class
name="Style"

Text: Role
name="text"
lowerBound=0
upperBound=-1

text: Cardinality
min=0
max=-1

style: Cardinality
min=1
max=1

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion

lirmm-00616272, version 1 - 21 Aug 2011
Propagation of the machining
The AnchorPrompt approach Adaptation

\[ W(x, y) = 1 - \left| \frac{\text{index}(x)}{\text{length}(X) - 1} - \frac{\text{index}(y)}{\text{length}(Y) - 1} \right| \]
Similarities propagation

Poetry:Entity
  name="Poetry"

poetry:Role
  name="poetry"

isA2:Relationship
  name="is a"

text2:Role
  name="text"

Text:Entity
  name="Text"

text:Role
  name="text"

has a:Relationship
  name="has a"

text:Property
  name="text"
  lowerBound=0
  upperBound=-1

has a:Association
  name="has a"

Poetry:Class
  name="Poetry"

g2:Generalization

Text:Class
  name="Text"
Similarities propagation

Poetry: Entity
  name="Poetry"

poetry: Role
  name="poetry"

isA2: Relationship
  name="is a"

Text: Entity
  name="Text"

has a: Relationship
  name="has a"

poetry: Role
  name="poetry"

Text: Entity
  name="Text"

text: Role
  name="text"

has a: Relationship
  name="has a"

Text: Entity
  name="Text"

text: Role
  name="text"

G2: Generalization

Text: Entity
  name="Text"

text: Property
  name="text"
  lowerBound=0
  upperBound=-1

has a: Association
  name="has a"
Empirical results

Precision calculation

![Precision calculation graph]

- **precision substring**
- **precision anchorPrompt**
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
Transformation rules generation

Examples

Models Matching

Transformation generation

Transformation rules

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

UML model

conforming to

UML metamodel

conforming to

Entity-Relationship model

conforming to

Entity-Relationship metamodel
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

- metaClassA : Property
  - title
  - firstName
  - lastName
  - year

- type : Concept_41
  - style

Property

Class
Lattice of model element
Target model (Entity-Relationship)
Lattice interpretation

rules lattice (excerpt)
Rules examples
Hypothesis

- closed world
- needs a good cover of the transformation

Different kinds of characteristics

- needed characteristics
- allowed characteristics
- forbidden characteristics
Forbidden characteristics

rules_11 - Concept_81

Property

Class

Attribute

Entity

entity [0..*]

type [0..0]

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

Concept_49 (S: 1)
type : Concept_41
elementA_16

Introduction FCA RCA By Example Example Matching Transformation Generation Conclusion
## Empirical study

<table>
<thead>
<tr>
<th>name</th>
<th>size of MapLinks lattice</th>
<th>number of extracted rules</th>
<th>premise size min</th>
<th>premise size max</th>
<th>premise size avg.</th>
<th>premise size std. dev.</th>
<th>premise depth min</th>
<th>premise depth max</th>
<th>premise depth avg.</th>
<th>premise depth std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>associations-persons</td>
<td>9</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>1.53</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0.82</td>
</tr>
<tr>
<td>Book2Publication</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1.21</td>
</tr>
<tr>
<td>delegation1</td>
<td>47</td>
<td>12</td>
<td>15</td>
<td>33</td>
<td>18</td>
<td>5.89</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>1.22</td>
</tr>
<tr>
<td>delegation2</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>17</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>0.87</td>
</tr>
<tr>
<td>Disaggregation</td>
<td>22</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>1.26</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>0.8</td>
</tr>
<tr>
<td>Ecore2Class</td>
<td>14</td>
<td>8</td>
<td>7</td>
<td>15</td>
<td>8</td>
<td>3.16</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>0.95</td>
</tr>
<tr>
<td>EliminateRedundantInheritance</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>11</td>
<td>8</td>
<td>1.63</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>1.27</td>
</tr>
<tr>
<td>emf2km3</td>
<td>26</td>
<td>19</td>
<td>14</td>
<td>52</td>
<td>20</td>
<td>9.48</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>0.77</td>
</tr>
<tr>
<td>EquivalenceAttributesAssociations</td>
<td>47</td>
<td>18</td>
<td>11</td>
<td>36</td>
<td>15</td>
<td>7.96</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>0.77</td>
</tr>
<tr>
<td>extractClass</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>1.15</td>
</tr>
<tr>
<td>Families2Persons_final</td>
<td>11</td>
<td>8</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>1.8</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1.22</td>
</tr>
<tr>
<td>hideDelegate</td>
<td>53</td>
<td>22</td>
<td>17</td>
<td>117</td>
<td>35</td>
<td>28.7</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>IntroducePrimaryKey</td>
<td>31</td>
<td>17</td>
<td>11</td>
<td>18</td>
<td>12</td>
<td>2.5</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>0.91</td>
</tr>
<tr>
<td>IntroducingInterface</td>
<td>38</td>
<td>11</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>3.02</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>1.61</td>
</tr>
<tr>
<td>JavaSource2Table</td>
<td>17</td>
<td>10</td>
<td>5</td>
<td>9</td>
<td>7</td>
<td>1.3</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>0.87</td>
</tr>
<tr>
<td>KM32EMF</td>
<td>18</td>
<td>15</td>
<td>11</td>
<td>20</td>
<td>14</td>
<td>3.74</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>1.35</td>
</tr>
<tr>
<td>KM32Problem</td>
<td>37</td>
<td>28</td>
<td>28</td>
<td>378</td>
<td>52</td>
<td>65.36</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>0.65</td>
</tr>
<tr>
<td>uml-er</td>
<td>35</td>
<td>24</td>
<td>2</td>
<td>11</td>
<td>6</td>
<td>2.75</td>
<td>2</td>
<td>9</td>
<td>5</td>
<td>1.15</td>
</tr>
<tr>
<td>uml-er-iccs</td>
<td>21</td>
<td>12</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td>1.78</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>1.54</td>
</tr>
<tr>
<td>uml-er2</td>
<td>25</td>
<td>15</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>2.19</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>1.41</td>
</tr>
<tr>
<td>uml-er3</td>
<td>40</td>
<td>19</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>0.83</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0.97</td>
</tr>
<tr>
<td>UML2ClassDiagram_to_KM3</td>
<td>200</td>
<td>40</td>
<td>28</td>
<td>187</td>
<td>58</td>
<td>40.51</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>2.72</td>
</tr>
</tbody>
</table>
### Empirical study

<table>
<thead>
<tr>
<th>nom</th>
<th>conclusion size</th>
<th>conclusion depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>associations-persons</td>
<td>nom</td>
<td>min</td>
</tr>
<tr>
<td></td>
<td>conclusion min</td>
<td>max</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Book2Publication</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>delegation1</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>delegation2</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Disaggregation</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>Ecore2Class</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>EliminateRedundantInheritance</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>emf2km3</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>EquivalenceAttributesAssociations</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>extractClass</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Families2Persons_final</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>hideDelegate</td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>IntroducePrimaryKey</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>IntroducingInterface</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>JavaSource2Table</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>KM32EMF</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>KM32Problem</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>uml-er</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>uml-er-iccs</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>uml-er2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>uml-er3</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>UML2ClassDiagram_to_KM3</td>
<td></td>
<td>18</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
Contribution

- Implementation independent with transformation metamodels
- Model matching assisted
- Result usable thanks to lattices
- Result genericity
Perspectives

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
Lopes et al. (2005)
Generating transformation definition from mapping specification: Application to web service platform. 

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.