Integration of Conceptual Data Modelling Languages

Pablo R. Fillottrani\(^1\)

joint work with Maria Keet\(^2\)

\(^1\)Depto. Cs. e Ing. de la Computación
Universidad Nacional del Sur
Bahía Blanca, Argentina

\(^2\)Department of Computer Science
University of Cape Town, South Africa

UCT, March 2015
1. Introduction

2. Metamodel

3. Conceptual Modelling Practice Analysis

4. Conclusions
Integration of Conceptual Data Modelling Languages

1. Introduction
2. Metamodel
3. Conceptual Modelling Practice Analysis
4. Conclusions
Preliminaries

- Bahía Blanca, Argentina
- Dept. of Computer Science and Engineering, Universidad Nacional del Sur
Preliminaries

- Bahía Blanca, Argentina
- Dept. of Computer Science and Engineering, Universidad Nacional del Sur
Preliminaries

- Bahía Blanca, Argentina
- Dept. of Computer Science and Engineering, Universidad Nacional del Sur
Introduction

Present our work in the bilateral AR-ZA project (2012-2014)

- Conceptual modelling
- Languages for conceptual modelling
- Develop formal basis for model integration tools and techniques
present our work in the bilateral AR-ZA project (2012-2014)
conceptual modelling
languages for conceptual modelling
develop formal basis for model integration tools and techniques
Introduction

- present our work in the bilateral AR-ZA project (2012-2014)
- conceptual modelling
  - languages for conceptual modelling
  - develop formal basis for model integration tools and techniques
Introduction

- present our work in the bilateral AR-ZA project (2012-2014)
- conceptual modelling
- languages for conceptual modelling
- develop formal basis for model integration tools and techniques
present our work in the bilateral AR-ZA project (2012-2014)
conceptual modelling
languages for conceptual modelling
develop formal basis for model integration tools and techniques
Context

- applications of conceptual modelling
- conceptual modelling language families:
  - EER
  - ORM
  - UML class diagrams
- conceptual modelling tools
Context

- applications of conceptual modelling
- conceptual modelling language families:
  - EER
  - ORM
  - UML class diagrams
- conceptual modelling tools
applications of conceptual modelling
conceptual modelling language families:
  - EER
  - ORM
  - UML class diagrams
conceptual modelling tools
Context

- applications of conceptual modelling
- conceptual modelling language families:
  - EER
  - ORM
  - UML class diagrams
- conceptual modelling tools
applications of conceptual modelling

conceptual modelling language families:
  - EER
  - ORM
    - UML class diagrams
  - conceptual modelling tools
applications of conceptual modelling
conceptual modelling language families:
  - EER
  - ORM
  - UML class diagrams
conceptual modelling tools
applications of conceptual modelling
conceptual modelling language families:
- EER
- ORM
- UML class diagrams
conceptual modelling tools
Previous work

- ICOM tool: project headed by Enrico Franconi, FUB, Italy
  
  http://www.inf.unibz.it/~franconi/icom/
Previous work

- ICOM tool: project headed by Enrico Franconi, FUB, Italy

  http://www.inf.unibz.it/~franconi/icom/
Previous work

- ICOM tool: project headed by Enrico Franconi, FUB, Italy
  
  [http://www.inf.unibz.it/~franconi/icom/](http://www.inf.unibz.it/~franconi/icom/)
ICOM methodology concept

- graphically design and integrate multiple ontology, with inter-ontology assertions
- complete logical reasoning support, not only to verify properties, but to show implicit facts and devise stricter constraints
- pluggable DL reasoner
- graphic language can express full ALCQI in an intuitive manner
ICOM methodology concept

- graphically design and integrate multiple ontology, with inter-ontology assertions
- complete logical reasoning support, not only to verify properties, but to show implicit facts and devise stricter constraints
- pluggable DL reasoner
- graphic language can express full ALCQI in an intuitive manner
ICOM methodology concept

- graphically design and integrate multiple ontology, with inter-ontology assertions
- complete logical reasoning support, not only to verify properties, but to show implicit facts and devise stricter constraints
  - pluggable DL reasoner
  - graphic language can express full ALCQI in an intuitive manner
ICOM methodology concept

- graphically design and integrate multiple ontology, with inter-ontology assertions
- complete logical reasoning support, not only to verify properties, but to show implicit facts and devise stricter constraints
- pluggable DL reasoner
- graphic language can express full ALCQI in an intuitive manner
ICOM methodology concept

- graphically design and integrate multiple ontology, with inter-ontology assertions
- complete logical reasoning support, not only to verify properties, but to show implicit facts and devise stricter constraints
- pluggable DL reasoner
- graphic language can express full ALCQI in an intuitive manner
Motivation

- apply conceptual modelling techniques to analyze conceptual modelling languages
- what’s behind them? is it possible to integrate them?
- check effectiveness of graphical syntax, need for reasoning support
Motivation

- apply conceptual modelling techniques to analyze conceptual modelling languages
- what’s behind them? is it possible to integrate them?
- check effectiveness of graphical syntax, need for reasoning support
Motivation

- apply conceptual modelling techniques to analyze conceptual modelling languages
- what’s behind them? is it possible to integrate them?
- check effectiveness of graphical syntax, need for reasoning support
Motivation

• apply conceptual modelling techniques to analyze conceptual modelling languages
• what’s behind them? is it possible to integrate them?
• check effectiveness of graphical syntax, need for reasoning support
Metamodel

captures all structural elements in the languages
also their relations and constraints
describes the rules in which they may be combined
the metamodel is formalized in FOL and OWL
Metamodelling

- captures all structural elements in the languages
- also their relations and constraints
- describes the rules in which they may be combined
- the metamodel is formalized in FOL and OWL
Metamodel

- captures all structural elements in the languages
- also their relations and constraints
- describes the rules in which they may be combined
- the metamodel is formalized in FOL and OWL
Metamodel

- captures all structural elements in the languages
- also their relations and constraints
- describes the rules in which they may be combined
- the metamodel is formalized in FOL and OWL
Metamodel captures all structural elements in the languages
also their relations and constraints
describes the rules in which they may be combined
the metamodel is formalized in FOL and OWL
Static entities

- basic building blocks of models
Introduction
Metamodel
Conceptual Modelling Practice Analysis
Conclusions

Static entities

- basic building blocks of models

Entity
Role
Relationship
Entity type
Constraint

- Entity
- Role
- Relationship
- Entity type
- Constraint

- Qualified relationship
- PartWhole
- Attributive property
- Subsumption
- Value property
- Data type
- Object type
- Qualifier

- Shared Aggregate
- Composite Aggregate
- Attribute
- Composite attribute
- Dimensional attribute
- Value type
- Dimensional value type
- Weak object type
- Nested object type
- Associative object type

- Multivalued attribute
- Mapped to
- Disjointness axioms among the subclasses of Relationship are:
  - {PartWhole, Attributive property, Subsumption}
  - {Qualified relationship, Attributive property, Subsumption}
Constraints

- specify properties of entities
specify properties of entities
Combination rules

- Specify how entities and constraints can be related
Combination rules

specify how entities and constraints can be related

A Weak identification is a combination of one or more Attributive property of the Weak object type it identifies together with the Identification constraint of the Object type it has a Relationship with and this Object type is disjoint with the Weak object type.

* The Single identification has a Mandatory and a 1:1 Cardinality constraint.
* Qualified identification and External identification are declared on only Attributive property.
* A Qualified relationship participates in a Qualified identification only if the Cardinality constraint is 1.
Transformation Rules

- a process for linking and translating models
- based on different kinds of rules: mappings, transformations, approximations
- together with the metamodel, it can be used to verify inter-model assertions
Transformation Rules

- a process for linking and translating models
- based on different kinds of rules: mappings, transformations, approximations
- together with the metamodel, it can be used to verify inter-model assertions
Transformation Rules

- a process for linking and translating models
- based on different kinds of rules: mappings, transformations, approximations
- together with the metamodel, it can be used to verify inter-model assertions
Transformation Rules

- a process for linking and translating models
- based on different kinds of rules: mappings, transformations, approximations
- together with the metamodel, it can be used to verify inter-model assertions
Transformation Rules
Transformation Rules

- take an entity, follow the sequence of mandatory constraints of the metamodel to transform using the algorithms containing the rules. repeat;
- process the remainder;
- ask user input for each approximation;
- record which are 1:1, remodelled, approximated, lost;

input model in language X

formalised metamodel

vocabulary containing a terminology comparison between terms used in the languages

output model in language Y

1:1 mappings
UML class ‘Flower’ -> ORM Entity Type ‘Flower’
Transformations
UML attribute ‘colour’ -> ORM Value Type ‘colour’
Integration of Conceptual Data Modelling Languages
Is expressiveness everything?

- very few elements belong to the three languages
- is it worth trying to integrate their models?
- we collect available models on each language, and study the usage of metamodel elements on them (approx. 35 on each language)
Is expressiveness everything?

- very few elements belong to the three languages
- is it worth trying to integrate their models?
- we collect available models on each language, and study the usage of metamodel elements on them (approx. 35 on each language)
Is expressiveness everything?

- very few elements belong to the three languages
- is it worth trying to integrate their models?
- we collect available models on each language, and study the usage of metamodel elements on them (approx. 35 on each language)
Is expressiveness everything?

- very few elements belong to the three languages
- is it worth trying to integrate their models?
- we collect available models on each language, and study the usage of metamodel elements on them (approx. 35 on each language)
Common Features

- classes (object types)
- attributes (or value type transformations)
- binary relationships
- class subsumption
- cardinality constraints on roles
- mandatory constraints
- single attribute identification
- all these elements represent more than 87% of use of all elements
Common Features

- classes (object types)
- attributes (or value type transformations)
- binary relationships
- class subsumption
- cardinality constraints on roles
- mandatory constraints
- single attribute identification
- all these elements represent more than 87% of use of all elements
Common Features

- classes (object types)
- attributes (or value type transformations)
- binary relationships
- class subsumption
- cardinality constraints on roles
- mandatory constraints
- single attribute identification
- all these elements represent more than 87% of use of all elements
Common Features

- classes (object types)
- attributes (or value type transformations)
- binary relationships
- class subsumption
- cardinality constraints on roles
- mandatory constraints
- single attribute identification
- all these elements represent more than 87% of use of all elements
Common Features

- classes (object types)
- attributes (or value type transformations)
- binary relationships
- class subsumption
  - cardinality constraints on roles
  - mandatory constraints
  - single attribute identification
- all these elements represent more than 87% of use of all elements
Common Features

- classes (object types)
- attributes (or value type transformations)
- binary relationships
- class subsumption
- cardinality constraints on roles
- mandatory constraints
- single attribute identification
- all these elements represent more than 87% of use of all elements
Common Features

- classes (object types)
- attributes (or value type transformations)
- binary relationships
- class subsumption
- cardinality constraints on roles
- mandatory constraints
- single attribute identification

all these elements represent more than 87% of use of all elements
Common Features

- classes (object types)
- attributes (or value type transformations)
- binary relationships
- class subsumption
- cardinality constraints on roles
- mandatory constraints
- single attribute identification

All these elements represent more than 87% of use of all elements
Common Features

- classes (object types)
- attributes (or value type transformations)
- binary relationships
- class subsumption
- cardinality constraints on roles
- mandatory constraints
- single attribute identification
- all these elements represent more than 87% of use of all elements
Salient Features of Each Family

- **UML**: object oriented, almost only binary relationships, no key, can express some constraints, but very few are used, relatively more use of isas

- **ORM**: relationship oriented, n-ary relationships, hidden attributes and less used, a lot of constraints can be expressed but rarely used (except for cardinalities), mainly single attribute identification

- **EER**: database oriented, n-ary relationships, different attributes and kinds of keys are more often used
Salient Features of Each Family

- **UML**: object oriented, almost only binary relationships, no key, can express some constraints, but very few are used, relatively more use of isas
- **ORM**: relationship oriented, n-ary relationships, hidden attributes and less used, a lot of constraints can be expressed but rarely used (except for cardinalities), mainly single attribute identification
- **EER**: database oriented, n-ary relationships, different attributes and kinds of keys are more often used
Salient Features of Each Family

- **UML**: object oriented, almost only binary relationships, no key, can express some constraints, but very few are used, relatively more use of isas.

- **ORM**: relationship oriented, n-ary relationships, hidden attributes and less used, a lot of constraints can be expressed but rarely used (except for cardinalities), mainly single attribute identification.

- **EER**: database oriented, n-ary relationships, different attributes and kinds of keys are more often used.
Salient Features of Each Family

- **UML**: object oriented, almost only binary relationships, no key, can express some constraints, but very few are used, relatively more use of isas
- **ORM**: relationship oriented, n-ary relationships, hidden attributes and less used, a lot of constraints can be expressed but rarely used (except for cardinalities), mainly single attribute identification
- **EER**: database oriented, n-ary relationships, different attributes and kinds of keys are more often used
Integration of Conceptual Data Modelling Languages

1. Introduction
2. Metamodel
3. Conceptual Modelling Practice Analysis
4. Conclusions
Integration of Conceptual Data Modelling Languages

1. Introduction
2. Metamodell
3. Conceptual Modelling Practice Analysis
4. Conclusions
conceptual modelling languages share enough common features to make integration feasible.

- modern tools for that must support graphic representation and integrate reasoning tasks.
- expressive power is heavily unused.
- future work: integrate these results into design tools.
conceptual modelling languages share enough common features to make integration feasible

- modern tools for that must support graphic representation and integrate reasoning tasks
- expressive power is heavily unused
- future work: integrate these results into design tools
Conclusions

- Conceptual modelling languages share enough common features to make integration feasible.
- Modern tools for that must support graphic representation and integrate reasoning tasks.
- Expressive power is heavily unused.
- Future work: integrate these results into design tools.
Conclusions

- Conceptual modelling languages share enough common features to make integration feasible.
- Modern tools for that must support graphic representation and integrate reasoning tasks.
- Expressive power is heavily unused.
- Future work: integrate these results into design tools.
conceptual modelling languages share enough common features to make integration feasible

modern tools for that must support graphic representation and integrate reasoning tasks

expressive power is heavily unused

future work: integrate these results into design tools
Thank you!!

Thank you!

- C. Maria Keet, Pablo Rubén Fillottrani: Toward an Ontology-Driven Unifying Metamodel for UML Class Diagrams, EER, and ORM2. ER 2013: 313-326
- C. Maria Keet, Pablo R. Fillottrani: Structural Entities of an Ontology-Driven Unifying Metamodel for UML, EER, and ORM2. MEDI 2013: 188-199
Thank you!

- C. Maria Keet, Pablo Rubén Fillottrani: Toward an Ontology-Driven Unifying Metamodel for UML Class Diagrams, EER, and ORM2. ER 2013: 313-326
- C. Maria Keet, Pablo R. Fillottrani: Structural Entities of an Ontology-Driven Unifying Metamodel for UML, EER, and ORM2. MEDI 2013: 188-199