Towards a Traceability Framework for Model Transformations in Kermeta

Jean-Rémy Falleri, Marianne Huchard, Clémentine Nebut

Towards a traceability framework for model transformations in Kermeta

Jean-Rémy Falleri, Marianne Huchard and Clémentine Nebut

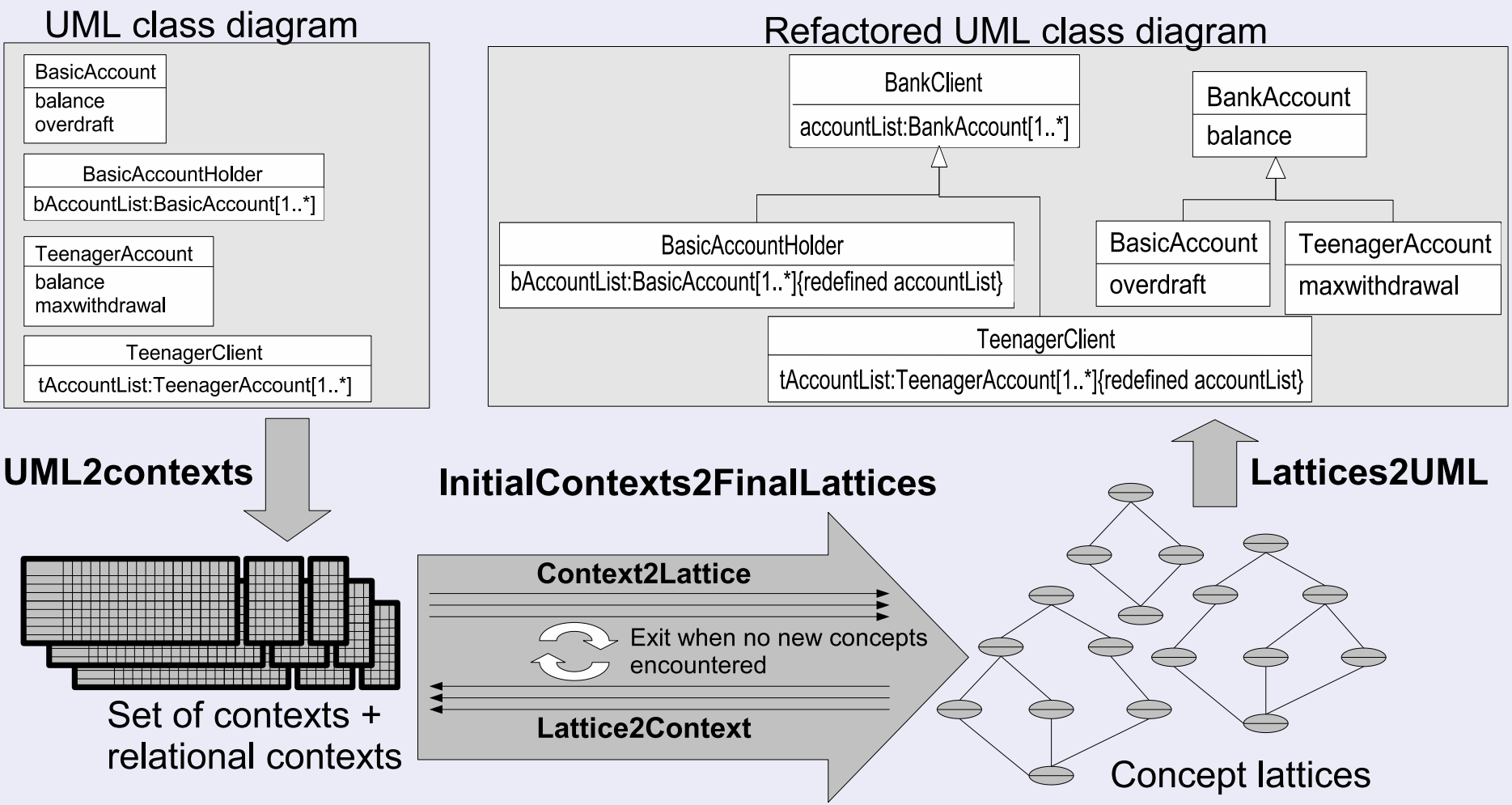
LIRMM, CNRS and Université de Montpellier 2

9 juillet 2006

Outline

Introduction

Context



Introduction

Context

- Definition of complex model transformations
- Usage of Kermeta language (imperative syntax)
- Transformations using information from previously applied transformations

Introduction

Context

- Definition of complex model transformations
- Usage of Kermeta language (imperative syntax)
- Transformations using information from previously applied transformations

Goals

- Need for a traceability framework

Kermeta

Description

- Developped at the *IRISA/INRIA* (*Triskell* Team), Rennes, France
- Object-oriented language
- Object model close to *MOF*
- Meta reflection
- Definition of structure and operational semantics of metamodels
- Integrated in *Eclipse*, compatible with *EMF*

Drawbacks

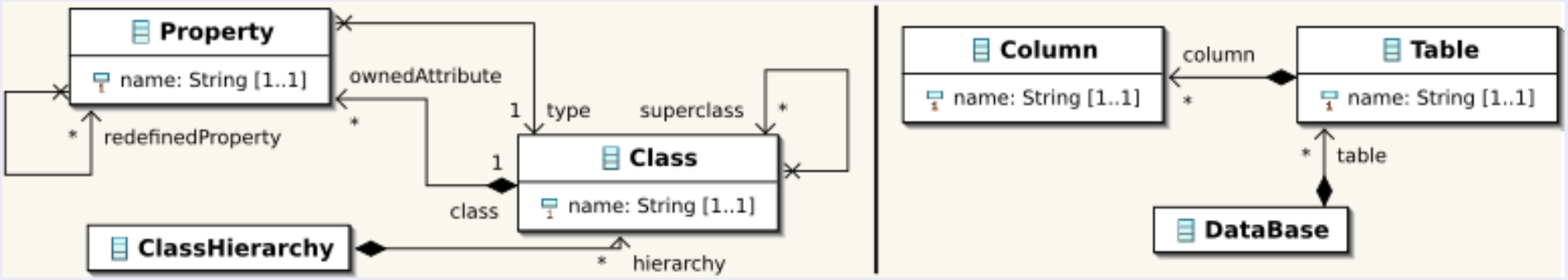
- No natural way to trace model transformations
- Impossible to avoid trace generating code

Example

Principe

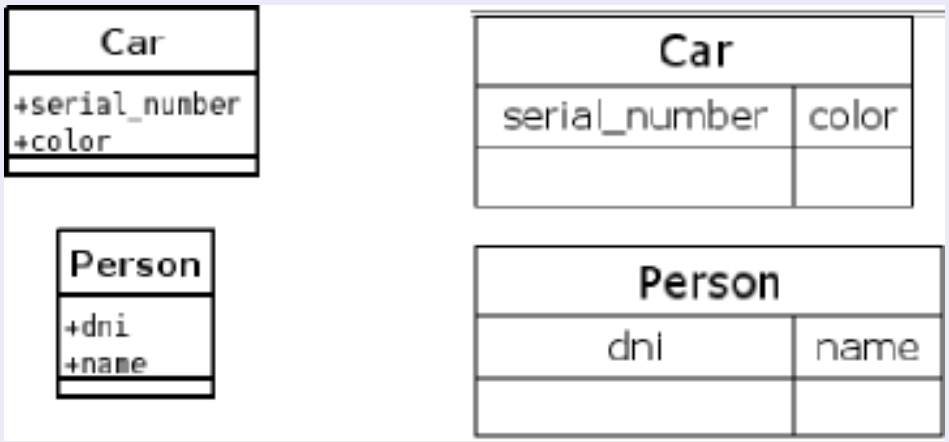
A *UML* class model is transformed into a *Database*

Metamodels

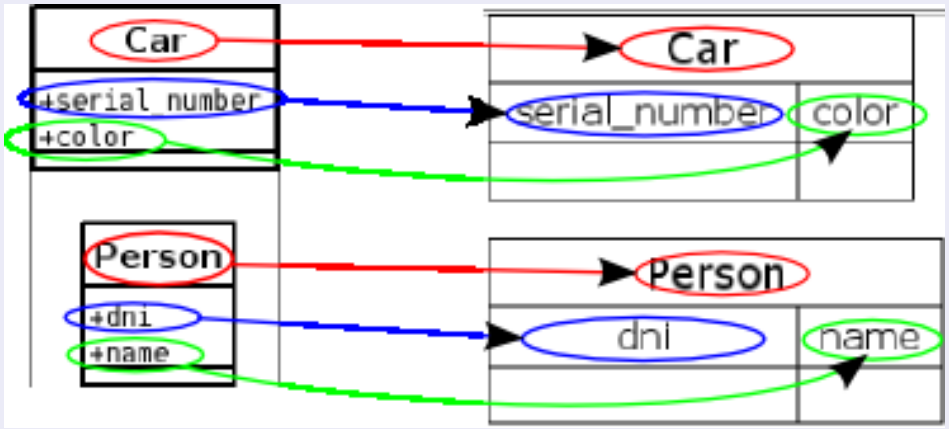


Example

Models



Trace

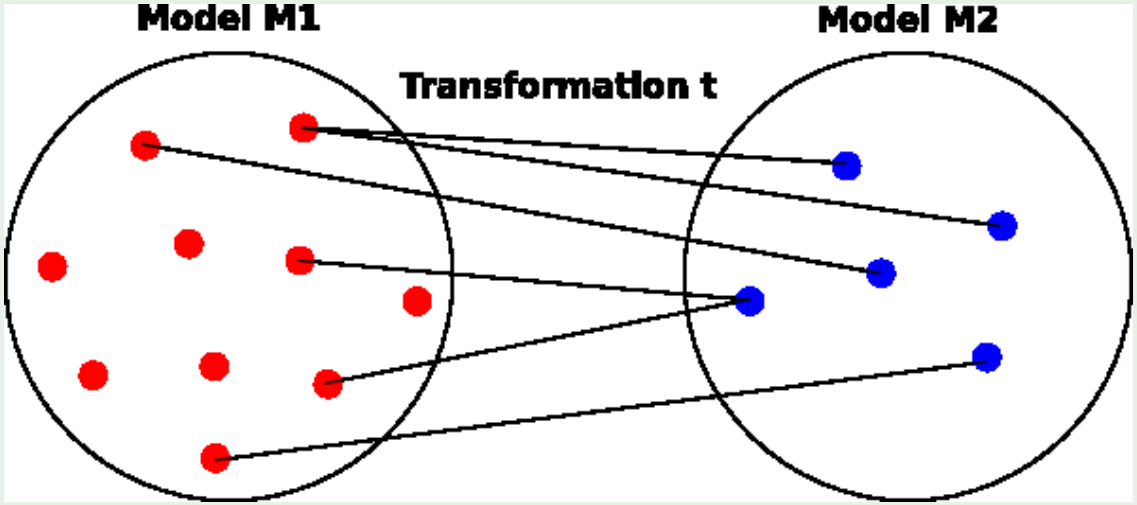


An intuitive model definition

Definition

- A model M is a set of elements.
- Let M1 and M2 be two models. A model transformation is a relation $t, t \subseteq M1 \times M2$.

Example

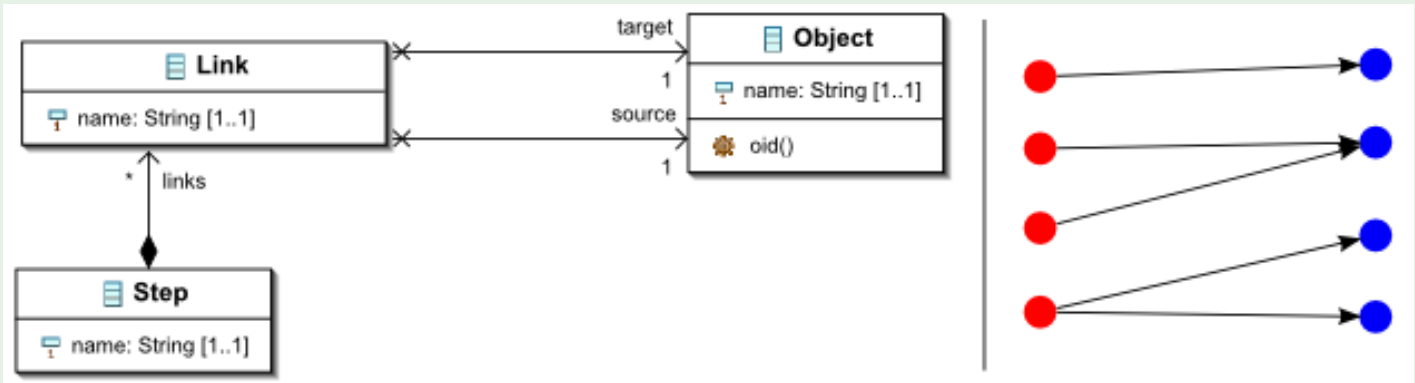


Trace definition

Definition

A step of a trace is a bipartite graph. The nodes are partitioned into two categories : source nodes and target nodes.

Example

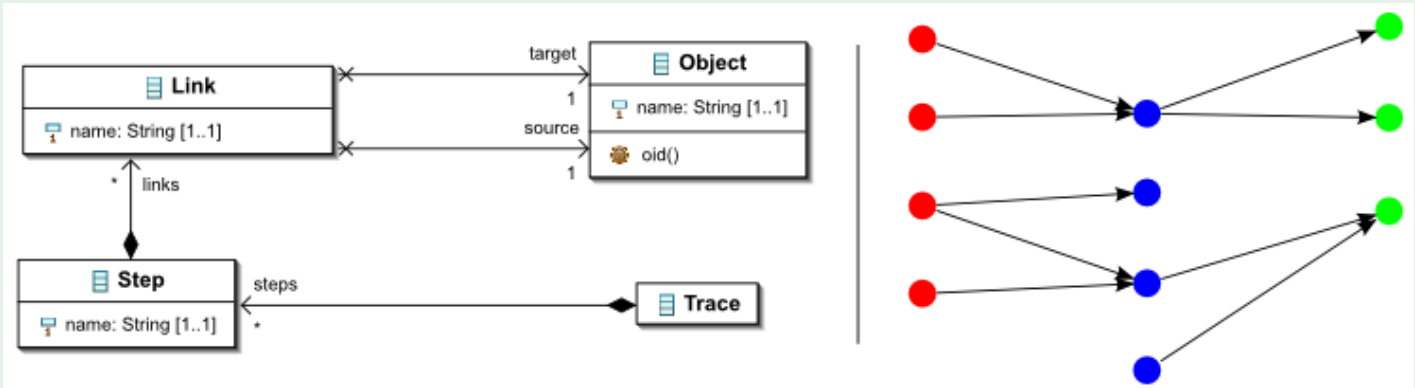


Trace definition

Definition

A transformation trace is a set of bipartite graphs.

Example

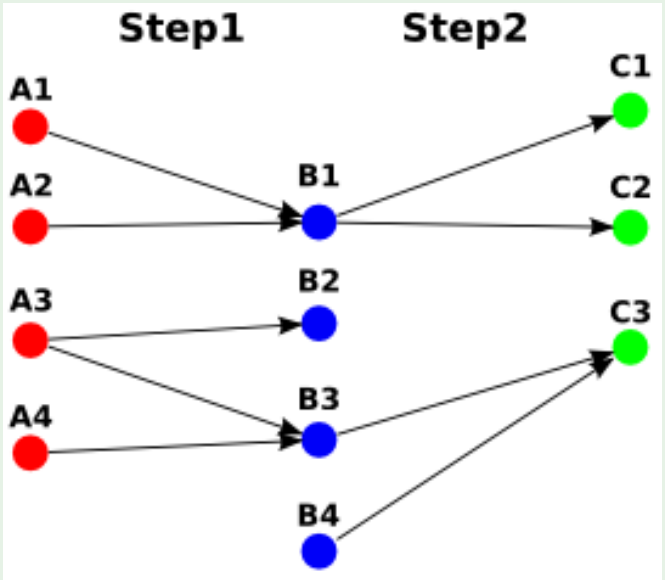


Basic operations

Definition

- $parents(C3) = \{B3, B4\}$
- $allparents(C3) = \{B3, B4, A4, A3\}$

Example



Implementation

Features

- Trace serialization (in *XMI* 2.0, thanks to *EMF*)
- Simple transformation from a trace to Graphviz's dot language (*AT-T*), in order to allow trace visualization

Constraints

- Trace generating code should be as short as possible, and only a small part of it should be placed in the transformation code
- Developers must be able to access to the elements of their choice through the trace
- Developers must be able to select the elements they want to trace

Serialization

Problems

- An element can't be contained by two elements
- During execution, developers need references

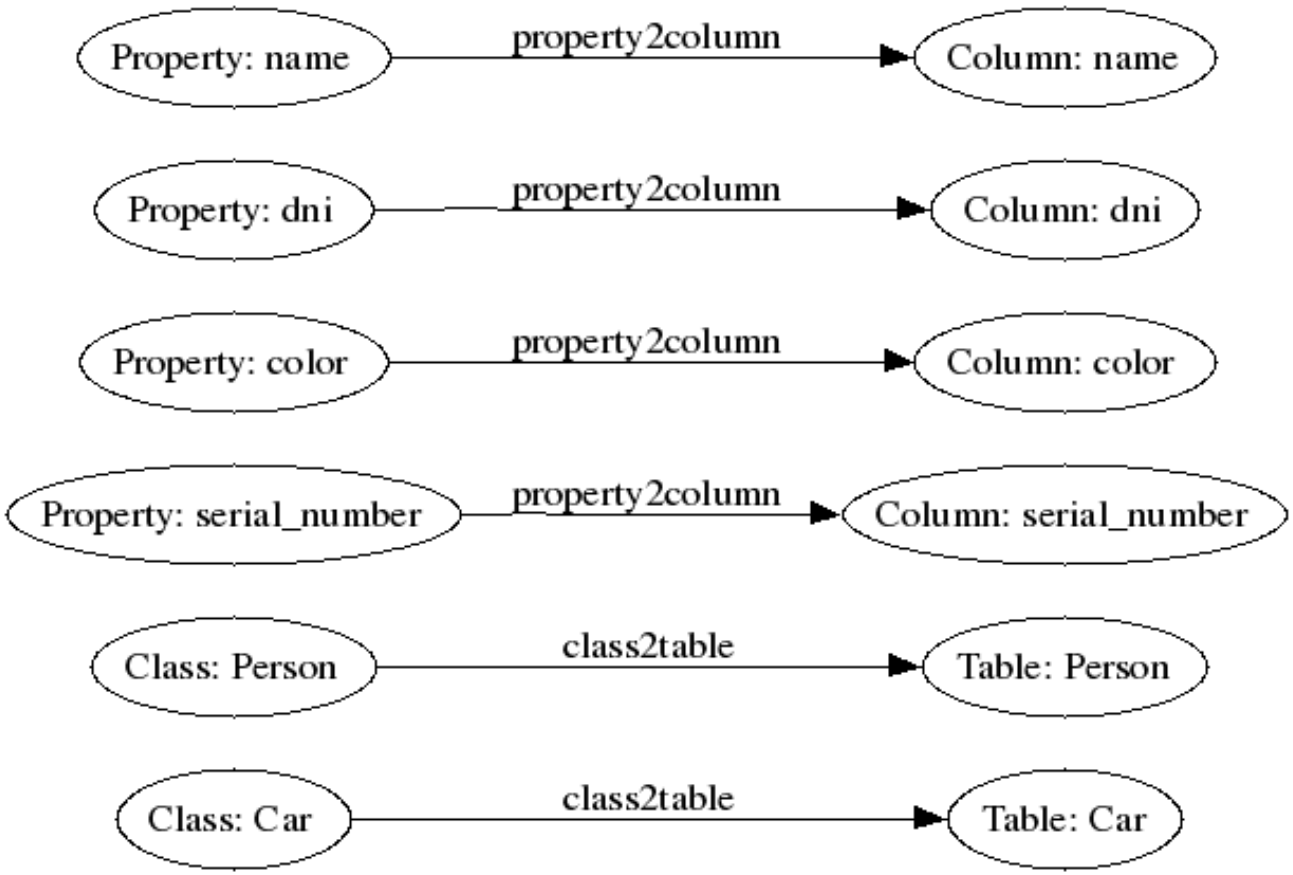
Current approach

- Use of references of model elements during execution
- Elements referenced are stored in the models
- Reduced clones are stored in the serialized trace

Transformation code

```
/**
 * Transform a minuml model to a mindb model
 */
operation transform(source: ClassHierarchy): DataBase is do
    result := DataBase.new // Initialize the target model

    trace.initStep("minuml2mindb") // Trace Generating Code

    source.hierarchy.each{ cls | // Iterate on every class of the source model
        var table: Table init Table.new // Create a Table
        table.name := String.clone(cls.name) // Copy the name of the Class to the table
        result.table.add(table) // Add the table in the target model

        trace.add_link("minuml2mindb","class2table",cls,table) // Trace Generating Code

        cls.ownedAttribute.each{ prop | // Iterate on every Property of the Class
            var col: Column init Column.new // Create a new Column
            col.name := String.clone(prop.name)
            table.column.add(col) // Add the Column to the relative Table

            trace.add_link("minuml2mindb","property2column",prop,col) // Trace Generating
                Code

        }
    }
end
```

Sample trace



Conclusion and future work

Conclusion

- An intuitive model definition
- A framework in the Kermeta language
- Difficulty to trace with an imperative syntax

Future work

- Composite links
- Graph-based trace definition

Open question

- Traceability is a cross-cutting concern
- Using aspects to cope with traceability in imperative transformation languages ?