Constraint-Aware Model Transformations

Adrian Rutle\textsuperscript{1}, Alessandro Rossini\textsuperscript{2}, Yngve Lamo\textsuperscript{1}, Uwe Wolter\textsuperscript{2}

\textsuperscript{1}Faculty of Engineering, Bergen University College, Norway
\textsuperscript{2}Department of Informatics, University of Bergen, Norway

22 March 2010
FASE 2010, Paphos, Cyprus
Introduction

Model transformation in model-driven engineering (MDE)
Introduction

Model transformation in model-driven engineering (MDE)
Introduction

Model transformation in model-driven engineering (MDE)

Our Approach

- Diagram Predicate Framework (DPF)
  - Integration of constraints in (meta)modelling
- Constraint-aware model transformation
  - Source constraints transformed to target constraints
  - Constraints used to control
    - which structure to create in the target model
    - which constraints to add to the created structure
• Model: abstract representation of a software system
• Req 1: “a person must be child of exactly two persons”
Modelling approach

- Model: specified by a modelling language

Sample model

OMG

Structural constraints

OMG
Modelling approach

- Modelling language: corresponding metamodel + semi-formal semantics
Modelling approach

- Req 2: “no person is his/her own child”
Modelling approach

- Modelling Language
  - Metamodel
    - Model
  - OCL Constraints

Sample model

- UML
  - Class
    - Property
      - Association
      - lower: Int
      - upper: Int
  - Person
    - childOf
      - context Person
        - inv: Irreflexive
        - self.ChildOf->excluding(self)

- Attached constraint: specified by means of text-based OCL expressions
Modelling approach

- Transformation rules: defined over the metamodel

Sample model

- UML
  - Class
  - Property
  - Association

- OCL
  - Structural constraints
  - Attached OCL constraints

- OMG
Introduction and Motivation

Modelling approach

- Transform. rule
- Metamodel
- OCL Constraints
- Modelling Language
- Model

- Specified by
- Defined over
- Conforms to
- Attached to

Sample model

- UML
- Property
  - lower: Int
  - upper: Int
- Association
- Person
  - childOf
  - inv: Irreflexive
  - self.ChildOf->excluding(self)

- Context Person

Challenge: transformation of attached constraints
Modelling approach

Modeling Language \(\xrightarrow{\text{corresp. to}}\) Metamodel \(\xrightarrow{\text{conforms to}}\) OCL Constraints \(\xrightarrow{\text{attached to}}\) Model

Transform. rule \(\xrightarrow{\text{defined over}}\) ?

Model Transformation Approach

Sample model

UML

Class \(\xrightarrow{\text{depends on}}\) Property \(\xrightarrow{\text{attached to}}\) Association

structural constraints

Attached OCL constraints

context Person

inv: Irreflexive

self.Child0f->excluding(self)

Model

Diagram Predicate Framework (DPF)
Modelling approach

- Specification: graph + set of constraints

Sample model

- UML
  - Class
  - Property
  - Association

- OCL
  - Structural constraints
  - Attached OCL constraints

- Context Person
  - Inv: Irreflexive
  - self. ChildOf ->excluding(self)
Modelling approach

- Modelling Language
  - Metamodel
    - Transform. rule
    - OCL Constraints
  - specified by
  - conforms to
  - attached to
  - defined over

Sample model

- UML
  - Class
    - Property
      - lower: Int
      - upper: Int
  - Association
  - Person
    - childOf
    - 2..2
    - context Person
      - inv: Irreflexive
      - self.ChildOf->excluding(self)

- Specification: graph + set of constraints
**Modelling approach**

- **Modelling Language**
  - **Transform. rule**
    - defined over
  - **Metamodel**
    - corresp. to
    - specifies by
    - conforms to
    - attached to
  - **Model**
    - OCL Constraints

- **Specification**
  - specified by a modelling formalism

**Sample model**

- **UML**
  - **Class**
  - **Property**
    - lower: Int
    - upper: Int
  - **Association**
    - Structural constraints
    - Attached OCL constraints

- **Person**
  - childOf
  - inv: Irreflexive
  - self.ChildOf->excluding(self)

- **OMG**
  - DPF
  - OCL
Modelling approach

- **Modelling formalism**: meta-specification + ...

---

Sample model

- **Context**: Person
  - **Property**: inv: Irreflexive
    - **OCL**: self.ChildOf->excluding(self)
  - **Class**: Person
  - **Association**
  - **Reference**
Introduction and Motivation

Model Transformation Approach

Summary and Future Work

Modelling approach

- Modelling formalism: meta-specification + signature

Sample model

OMG DPF

UML

Class

Property

Association

lower: Int

upper: Int

context Person

inv: Irreflexive

self.ChildOf->excluding(self)

OMG DPF

Class

Reference

Diagrammatic signature

$\forall x \in X : n \leq f(x) \leq m$

$\forall x \in X : x \notin f(x)$
Introduction and Motivation

Model Transformation Approach

Summary and Future Work

Modelling approach

- Modelling Language
  - Metamodel
  - OCL Constraints
    - Transform. rule
  - Model
    - Transform. rule
  - Metamodel
    - OCL Constraints
      - defined over
  - Modelling Language
    - Metamodelling
      - integrated constraints
  - Specification
    - specified by
  - Meta-specification
    - conforms to

Sample model

- UML
  - Class
    - Property
      - lower: Int
      - upper: Int
  - Association
    - Structural constraints
      - Attached OCL constraints
  - Person
    - childOf
      - context Person
        - inv: Irreflexive
        - self.ChildOf->excluding(self)
  - Class
    - Reference
      - Diagrammatic signature
        - \( p \) [intended semantics]
          - \( \text{[mult}(n,m)\} \): \( \forall x \in X : n \leq f(x) \leq m \)
          - \( \text{[irreflexive]} \): \( \forall x \in X : x \notin f(x) \)

- Integration of constraints
Modelling approach

- Transform. rule
- Metamodel
  - OCL Constraints
  - Model
  - Metamodel
    - Modelling Language
      - correspond to
        - specified by
          - Modelling Formalism
            - Metamodel
              - Signature
            - Specification
  - transforms over
    - defined over
      - Transform. rule

Sample model

- UML
  - Class
  - Property
    - lower: Int
    - upper: Int
  - Association
    - Structural constraints
      - context Person
        - inv: Irreflexive
        - self.ChildOf->excluding(self)
    - Attached OCL constraints
  - OCL
    - childOf

- Diagrammatic signature
  - $P$ | Intended semantics
  - $\forall x \in X : n \leq |f(x)| \leq m$
  - $\forall x \in X : x \notin f(x)$
Transformation rules: defined over the modelling formalism
Sample Object-Oriented Modelling Hierarchy

- Modelling formalism: meta-specification + ...
Sample Object-Oriented Modelling Hierarchy

- Modelling formalism: meta-specification + signature
Sample Object-Oriented Modelling Hierarchy

• Req 1: “an employee must work for at least one department.”
Sample Object-Oriented Modelling Hierarchy

- **Req 2**: “a department may have none or many employees”
Sample Object-Oriented Modelling Hierarchy

- Req 3: "a project may involve none or many employees"
Sample Object-Oriented Modelling Hierarchy

- Req 4: “a project must be controlled by at least one department”
• Req 5: “an employee involved in a project must work in the controlling department”
Sample Object-Oriented Modelling Hierarchy

\[\begin{array}{|c|c|c|c|}
\hline
p & \mathcal{\Sigma}_2(p) & \text{Proposed vis.} & \text{Intended semantics} \\
\hline
\text{[mult}(n,m)] & 1 \xrightarrow{f} 2 & \checkmark & \forall x \in X : n \leq |f(x)| \leq m \\
\hline
\text{[surjective]} & 1 \xrightarrow{f} 2 & \checkmark & f(X) = Y \\
\hline
\text{[inverse]} & 1 \circlearrowright 2 & \checkmark & \forall x \in X, \forall y \in Y : y \in f(x) \iff x \in g(y) \\
\hline
\text{[image-inclusion]} & 1 \xrightarrow{f} 2 & \checkmark & \forall x \in X : f(x) \subseteq g(x) \\
\hline
\text{[composition]} & 1 \xrightarrow{f} 2 \circlearrowright 3 & \checkmark & \forall x \in X : f ; g(x) = \{ g(y) \mid y \in f(x) \} \\
\hline
\end{array}\]
Sample Object-Oriented Modelling Hierarchy

- **Invalid instance**
Outline

Introduction and Motivation

Model Transformation Approach

Sample Model Transformation

Summary and Future Work
Model Transformation Approach

- Given source and target modelling formalisms and a source model ...

Source Modelling Formalism

specified by

Source Model

Target Modelling Formalism
... we want to generate a target model
Model Transformation Approach

- How we get the target model from the source model?
Model Transformation Approach

- We have to relate the modelling formalisms
Model Transformation Approach

- First: Define morphisms to an appropriate joined modelling formalism
• Second: Define constraint-aware transformation rules
Third: Apply the model transformation
Model Transformation Approach

- Third: Apply the model transformation
  - A: Convert the source model to an intermediate model
• Third: Apply the model transformation
  • B: Apply the transformation rules (Pushout)
Model Transformation Approach

- Third: Apply the model transformation
  - C: Project out the target model (Pullback)
Heterogeneous, out-place model transformation
Outline

Introduction and Motivation

Model Transformation Approach

Sample Model Transformation

Summary and Future Work
Step 1: Joining Modelling Formalisms
Step 1: Joining Modelling Formalisms

\[ \text{Class} \xrightarrow{\text{Reference}} \text{Attribute} \xrightarrow{\text{Data Type}} S_2 \]

\[ \text{Class} \xrightarrow{\text{Reference}} \text{Attribute} \xrightarrow{\text{Data Type}^a} \xrightarrow{[\text{bij}]} \xrightarrow{\text{Table}} J_2 \]

\[ \text{Table} \xrightarrow{\text{Column}^{[1..\infty]}} \text{Data Type} \]

\[ \text{Table} \xrightarrow{\text{Column}^{[1..\infty]}} \text{Data Type} \]
Step 2: Define Constraint-Aware Rules

- Rule $r_1$. Class to table and primary key
Step 2: Define Constraint-Aware Rules

- Rule $r_2$. Attribute to column

<table>
<thead>
<tr>
<th>$\mathcal{L}$</th>
<th>$\mathcal{R}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1:C$ $\rightarrow$ $1:T$ $\rightarrow$ Int:DT$^t$</td>
<td>$1:C$ $\rightarrow$ $1:T$ $\rightarrow$ Int:DT$^t$</td>
</tr>
<tr>
<td>$1:A$ $\rightarrow$ $1:DT^S$</td>
<td>$1:A$ $\rightarrow$ $2:Col$ $\rightarrow$ $1:DT^t$</td>
</tr>
</tbody>
</table>
Step 2: Define Constraint-Aware Rules

- Rule $r_3$. Many-to-one references to foreign key

```
<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image-url" alt="Diagram" /></td>
<td><img src="image-url" alt="Diagram" /></td>
</tr>
</tbody>
</table>
```
Step 2: Define Constraint-Aware Rules

- Rule $r_4$. Many-to-many references to link table and foreign key
Step 2: Define Constraint-Aware Rules

- Rule $r_5$. Many-to-many reference to link table and foreign keys
Step 2: Define Constraint-Aware Rules

- Rule $r_6$. [inverse] and [surjective] to [foreign-key], [image-equal] and [jointly-injective]

<table>
<thead>
<tr>
<th>$L$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule $r_6$. [inverse] and [surjective] to [foreign-key], [image-equal], [total] and [jointly-injective]</td>
<td>Rule $r_6$. [inverse] and [surjective] to [foreign-key], [image-equal], [total] and [jointly-injective]</td>
</tr>
</tbody>
</table>

![Diagram showing the definition of constraint-aware rules](image-url)
Step 3: Applying Model Transformation

Introduction and Motivation

Model Transformation Approach

Summary and Future Work
A: Type Conversion of Source Model
A: Type Conversion of Source Model
B: Applying the Rules (after $r_5$)

Blue: Matched, Green: Added
B: Applying the Rules (after $r_6$)

Blue: Matched, Green: Added
Last Intermediate Model
C: Projection
C: Projection
Outline

Introduction and Motivation

Model Transformation Approach
   Sample Model Transformation

Summary and Future Work
Summary

- Diagram Predicate Framework (DPF)
  - Integration of constraints in (meta)modelling
- Constraint-aware model transformation
  - Source constraints transformed to target constraints
  - Constraints used to control
    - which structure to create in the target model
    - which constraints to add to the created structure
Summary

- Diagram Predicate Framework (DPF)
  - Integration of constraints in (meta)modelling
- Constraint-aware model transformation
  - Source constraints transformed to target constraints
  - Constraints used to control
    - which structure to create in the target model
    - which constraints to add to the created structure
- Out-place approach to heterogeneous model transformation
Related work

- **Graph Transformation System (GTS)** *Hartmut Ehrig et al*
  - Adding support for transforming constraints
- **Triple Graph Grammar (TGG)** *Schürr et al*
  - Adding support for diagrammatic constraints in the joined modelling formalism
- **Refactoring** *Fondement and Baar*
  - Transformation of constraints in refactoring UML/OCL models
  - Adding support for a more general case
- **VTMS** *Lengyl and Levendovzsky*
  - Validation of transformations by means of OCL constraints
  - Puts constraints on rules to narrow down matches
  - A match is a metamodel instantiation
- **Analyses of GTS** *Cabot et al*
  - Rules and their properties e.g. conflicts and rule applicability to OCL expressions
Future work

- Controlling and scheduling transformation rules
- Morphisms between modelling formalisms: to appear in Journal of Logic and Algebraic Programming
- Prototype tool
- Comparison of the tool to other tools: AGG, GReAT, PROGRESS, VMTS, Epsilon, Fujaba etc
- Real-size case study: noark (Norwegian Archive standard)
Thank you!

Questions?
Challenges in modelling

- Mixing graph-based structures with textual constraints
  - Different technical spaces
    - checking models in two different engines/steps
    - model-constraint synchronisation problem
    - violation of “everything-is-a-model” vision of MDE
  - Challenge for domain experts who do not understand OCL
Formalisation approach

- Based on category theory
  - Sketches formalism: define semantics of diagrams (thus graph-based models)
    - models: graphs (nodes and edges)
    - model properties: universal properties (limit, colimit, commutative diagrams)
  - Generalized sketches formalism
    - not only universal properties
    - user-defined diagrammatic predicate signatures
- DPF: specification formalism based on generalized sketches
## Sample Signature for Relational Data Models

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\alpha^{\Theta_2}(p)$</th>
<th>Proposed vis.</th>
<th>Intended semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>[primary-key]</td>
<td>$\begin{array}{c} f \ 1 \rightarrow 2 \end{array}$</td>
<td>$\begin{array}{c} X \ \rightarrow f \ [pk] \ Y \end{array}$</td>
<td>$f$ is [total] and [injective]</td>
</tr>
<tr>
<td>[foreign-key]</td>
<td>$\begin{array}{c} f \ 1 \rightarrow 2 \end{array}$</td>
<td>$\begin{array}{c} X \ \rightarrow f \ [fk] \ Y \ g \end{array}$</td>
<td>$f(X) \subseteq g(Y)$</td>
</tr>
<tr>
<td>[image-equal]</td>
<td>$\begin{array}{c} f \ 1 \rightarrow 2 \end{array}$</td>
<td>$\begin{array}{c} X \ \rightarrow f \ [ie] \ Y \ g \ Z \end{array}$</td>
<td>$f(X) = g(Z)$</td>
</tr>
<tr>
<td>[join]</td>
<td>$\begin{array}{c} f \ 1 \rightarrow 2 \ g' \ 3 \rightarrow 4 \ f' \end{array}$</td>
<td>$\begin{array}{c} X \ \rightarrow f \ Y \ g' \end{array}$</td>
<td>$\forall x \in X, \forall z \in Z : (x, z) \in XZ$ iff $f(x) = g(z)$</td>
</tr>
</tbody>
</table>