Towards a Formal Approach to Metamodel Evolution

Florian Mantz\textsuperscript{1}, Alessandro Rossini\textsuperscript{2}, Adrian Rutle\textsuperscript{1}, 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

11 November 2010
Nordic Workshop on Programming Theory, Turku, Finland
Outline

Introduction and Motivation

Diagram Predicate Framework

Metamodel Evolution
Model-Driven Engineering (MDE)

- Model-driven engineering
  - Models as primary artefacts of software development
  - Model transformations
  - Metamodels as language descriptions
- Complex evolution of models and metamodels
  - Need for techniques and tools to support evolution
Metamodel Evolution
Metamodel Evolution

Metamodel

Model_x

Model_y

Metamodel

Metamodeĺ

evolution

conf

conf
Metamodelling Evolution

\[ \text{Metamodel} \xrightarrow{\text{evolution}} \text{Metamodel'} \]

\[ \text{Model}_x = \text{conf} \]

\[ \text{Model}_y = \text{conf} \]

\[ \text{Model'}_x = \text{migration} \]

\[ \text{Model'}_y = \text{migration} \]
Metamodel Evolution

\[
\text{Model}_x \xrightarrow{\text{conf}} \text{Model} \xrightarrow{\text{evolution}} \text{Model}^{\prime} \xrightarrow{\text{conf}} \text{Model}^{\prime}_x
\]

\[
\text{Model}_y \xrightarrow{\text{conf}} \text{Model} \xrightarrow{\text{migration}} \text{Model}^{\prime} \xrightarrow{\text{conf}} \text{Model}^{\prime}_y
\]

\[
\text{Metamodel} \xrightarrow{\text{confidence}} \text{Model} \xrightarrow{\text{migration}} \text{Model}^{\prime}
\]
Metamodel Evolution

\[
\text{Model}_x \xrightarrow{\text{conf}} \text{Model}_x' \xrightarrow{\text{conf}} \text{Model}_y \xrightarrow{\text{migration}} \text{Model}_y' \xrightarrow{\text{migration}} \text{Metamodel} \xrightarrow{\text{evolution}} \text{Metamodel}'
\]
• Metamodels have many dependencies!
Metamodel Evolution Challenges

- Description of metamodel evolution
- Migration of dependent artefacts
  - Models
  - Model transformations
  - Documentations
  - ...
- Lack of formalisation
Diagram Predicate Framework (DPF)

- Formal approach to MDE based on category theory
  - (Meta)modelling
  - Model transformation
  - Model management
- Models in DPF
  - Graph + set of diagrammatic constraints
- Metamodels in DPF
  - Graph + set of diagrammatic constraints
Pushouts & Pullbacks

- Pushouts are used for rule applications
- Pullbacks are used to restrict models
Outline

Introduction and Motivation

Diagram Predicate Framework

Metamodel Evolution
Sample GRID Metamodell

- Modelling language: metamodel + ...
Sample GRID Metamodel

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\alpha^{\Sigma^{2}}(p)$</th>
<th>Proposed vis.</th>
<th>Intended semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>[mult ($m, n$)]</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>$X \xrightarrow{f}_{[m..n]} Y$</td>
<td>$\forall x \in X : m \leq</td>
</tr>
<tr>
<td>[surjective]</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>$X \xrightarrow{f}_{[surj]} Y$</td>
<td>$f(X) = Y$</td>
</tr>
<tr>
<td>[irreflexive]</td>
<td></td>
<td></td>
<td>$\forall x \in X : x \neq f(x)$</td>
</tr>
<tr>
<td>[image-inclusion]</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>$X \xrightarrow{f}_{[\subseteq]} Y$</td>
<td>$\forall x \in X : f(x) \subseteq g(x)$</td>
</tr>
<tr>
<td>[composition]</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>$X \xrightarrow{f}_{[\text{comp}]} Y$</td>
<td>$\forall x \in X : h(x) = {g(y) \mid y \in f(x)}$</td>
</tr>
</tbody>
</table>

- Modelling language: metamodel + signature
Sample GRID Metamodel

- **Req 1:** “a process is owned exactly by one user”
- **Req 2:** “a process runs exactly on one computing node”
- **Req 3:** “a process cannot depend on itself”
Sample GRID Metamodel

- Models have to conform to their metamodel
**Sample GRID Metamodel**

<table>
<thead>
<tr>
<th>p</th>
<th>$\alpha^{X^2}(p)$</th>
<th>Proposed vis.</th>
<th>Intended semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[\text{mult}(m,n)]$</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>$X \xrightarrow{f} Y$</td>
<td>$\forall x \in X : m \leq</td>
</tr>
<tr>
<td>$[\text{surjective}]$</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>$X \xrightarrow{f} Y$</td>
<td>$f(X) = Y$</td>
</tr>
<tr>
<td>$[\text{irreflexive}]$</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>$X \xrightarrow{f} Y$</td>
<td>$\forall x \in X : x \neq f(x)$</td>
</tr>
<tr>
<td>$[\text{image-inclusion}]$</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>$X \xrightarrow{f} Y$</td>
<td>$\forall x \in X : f(x) \subseteq g(x)$</td>
</tr>
<tr>
<td>$[\text{composition}]$</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>$X \xrightarrow{f} Y$</td>
<td>$\forall x \in X : h(x) = {g(y) \mid y \in f(x)}$</td>
</tr>
</tbody>
</table>

New Constraint:

A Process $P$ running on ComputingNode $CN$ can only depend on a Process $P'$ running on ComputingNode $CN'$ if ComputingNode $CN'$ is reachable by ComputingNode $CN$.

- The metamodel may be revised, e.g. new constraints may be added
- This may break the conformance of some models
Sample GRID Metamodel

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\alpha^{k^2}(p)$</th>
<th>Proposed vis.</th>
<th>Intended semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>([\text{mult}(m,n)])</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>(X \xrightarrow{f} \gamma [\text{m}, n] )</td>
<td>$\forall x \in X : m \leq</td>
</tr>
<tr>
<td>([\text{surjective}])</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>(X \xrightarrow{f} \gamma [\text{sur}] )</td>
<td>$f(X) = Y$</td>
</tr>
<tr>
<td>([\text{irreflexive}])</td>
<td>$\circ f$</td>
<td>(X \xrightarrow{f} \gamma [\text{irr}] )</td>
<td>$\forall x \in X : f(x)$</td>
</tr>
<tr>
<td>([\text{image-inclusion}])</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>(X \xrightarrow{f} \gamma [\text{in}] )</td>
<td>$\forall x \in X : f(x) \subseteq g(x)$</td>
</tr>
<tr>
<td>([\text{composition}])</td>
<td>$1 \xrightarrow{a} 2$</td>
<td>(X \xrightarrow{f} \gamma [\text{comp}] )</td>
<td>$\forall x \in X : h(x) = {g(y) \mid y \in f(x)}$</td>
</tr>
</tbody>
</table>

New Constraint:

A Process P running on ComputingNode CN can only depend on a Process P* running on ComputingNode CN' if ComputingNode CN' is reachable by ComputingNode CN.

- Models have to be “repaired” in one way or another
Outline

Introduction and Motivation

Diagram Predicate Framework

Metamodel Evolution
Example

Metamodel
Example

Metamodel Evolution

Add new Constraint

New Constraint:

A Process $P$ running on ComputingNode $CN$ can only depend on a Process $P'$ running on ComputingNode $CN'$ if ComputingNode $CN'$ is reachable by ComputingNode $CN$. 
Example

Metamodel Evolution

Add new Constraint
Example

Metamodel Evolution

Define metamodel evolution by rule: pushout construction
Example

For any model $G_1$ conforming to the metamodel $G_2$
Example

For any model $\mathcal{G}_1$ conforming to the metamodel $\mathcal{G}_2$

What is the corresponding model $\mathcal{G}'_1$ conforming to the metamodel $\mathcal{G}'_2$?
**Example**

$I^\mathcal{L}$ can be calculated by pullback construction.

$I^\mathcal{R}$ can be constructed as a pullback complement.
Example

Migration rule $I^\mathcal{L} \rightarrow I^\mathcal{M}$ applied to the model $\mathcal{G}_1$ must create a commuting cube.
There may be other models conforming to the metamodel $S_2$
Example

The same metamodel $\mathcal{G}_2$ and a different model $\mathcal{G}_1$
Example

$I^\mathcal{L}$ can be calculated by pullback construction

$I^\mathcal{R}$ can be constructed as a pullback complement
Example

Migration rule $I^\mathcal{S} \rightarrow I^\mathcal{M}$ applied to the model $\mathcal{S}_1$ must create again a commuting cube.
Example

Need an equivalent rule that can migrate all models
Communting Cube

\[ \mathcal{L} \xrightarrow{m} \mathcal{S}_2 \xrightarrow{r'} \mathcal{S}'_2 \xrightarrow{r} \mathcal{R} \xrightarrow{m'} \mathcal{S}'_2 \]
• The top face is a pushout
Communting Cube

- The left face is a pullback
• The back face is a pullback
For a rule $r$ the semantics $\llbracket r \rrbracket$ can be any mapping satisfying the condition that the back face is a pullback.
• If the bottom face is a pushout
• Then the front face and the right face have to be pullbacks
• Need for a rule which captures all conforming models
The rule $r_s$ has to be minimal

- It has to complete the cube for any conforming model $\mathcal{G}_1$
The blue parallelogram has to commute

The yellow parallelogram has to be a pushout
Future Work

- Adapt the DPF approach to constraint-aware model transformation for all layers of a metamodelling hierarchy
- Find the conditions on the rules that guarantee a commuting cube
- Extend the cube for the case that one minimal rule is not sufficient
- Investigate constraints on the model level
- Investigate evolution of model transformations
Thank you!

Questions?

For more information and list of publications visit: http://dpf.hib.no/