A Formal Approach to Modeling and Model Transformations in Software Engineering

Adrian Rutle

Department of Computer Engineering
Faculty of Engineering
Bergen University College

17.06.2008
EXPO 2008, Integration and Architecture, Bergen, Norway
History of the Project

- **2003** Wolter and Lamo: decided to use Generalized Sketches approach in software engineering
- **2004** Ørjan Hatland (master student):
  - implementation of tools for GS
  - .NET framework
- **2006** Adrian Rutle (PhD student):
  - MDE, Eclipse platform
  - DPL
- **2006** Stian Skjerveggen (master student):
  - implementation of graphical editor (EMF/GMF)
- **2008** Thomas Sliåk Tvedt (master student):
  - serialization of diagrammatic specifications
  - continuation of Stian’s tools
Engineering techniques where models are first-class entities

- Evolved from the popularity of diagrammatic languages such as UML and ER and their popularity for specification and documentation of software systems
- Aims to raise the abstraction level of software development from text to models
- The development process is based on software models, not on code
Industry "standard" for MDE

- Model Driven Architecture (MDA) approach from Object Management Group (OMG) where:
  - First step is building platform independent domain models (PIM)
  - PIMs are refined to platform specific models (PSM)
  - Refinement process more or less automatized as model transformations
  - Code generation and model refinement can be seen as model transformations
  - Reuse of software models by using model transformations
Key-points for the Success of MDE

Models are no longer only for documentation

- Models should be **both**:
  - intuitive to ease their interception by *software engineers* and *domain experts*
  - formal such that *machines* can have a precise understanding of (and able to reason about) the models

⇒ Diagrammatic approach enhances human understanding
⇒ One challenge is to combine intuition with formalization
⇒ Another challenge is to implement the formalization approach
Models are no longer only for documentation

- Models should be both:
  - intuitive to ease their interception by software engineers and domain experts
  - formal such that machines can have a precise understanding of (and able to reason about) the models

⇒ Diagrammatic approach enhances human understanding
⇒ One challenge is to combine intuition with formalization
⇒ Another challenge is to implement the formalization approach
Key-points for the Success of MDE

Models are no longer only for documentation

- Models should be both:
  - intuitive to ease their interception by software engineers and domain experts
  - formal such that machines can have a precise understanding of (and able to reason about) the models

⇒ Diagrammatic approach enhances human understanding
⇒ One challenge is to combine intuition with formalization
⇒ Another challenge is to implement the formalization approach

Adrian Rutle
DPL, MDE and MT (http://gs.hib.no)
Models are no longer only for documentation

- Models should be **both**:
  - intuitive to ease their interception by *software engineers* and *domain experts*
  - formal such that *machines* can have a precise understanding of (and able to reason about) the models

⇒ Diagrammatic approach enhances human understanding
⇒ One challenge is to combine intuition with formalization
⇒ Another challenge is to implement the formalization approach
Some facts about UML

- UML is considered *de facto* industry standard for modeling
- UML can only express constraints on binary relations
- UML has serious issues regarding:
  - Semantics; UML models may be ambiguous and have semi-formal semantics
  - Complexity; UML uses 13 different types of diagrams
  - Expressibility; To express constraints over higher order relations one need to use string based logic, (Object Constraint Logic, OCL)
Some facts about Eclipse Modeling Framework

- EMF is considered a low-cost entry to the employment of model-driven engineering
- EMF is used increasingly by companies which adopt the Eclipse platform
- The metamodel of EMF is very close to Essential MOF
- Less complex ⇔ clearer semantics
- Not just a standard, but an implementation
- Uses XMI for serialization of models
Precise semantics, proven abilities and several fundamental Software Engineering problems solved by use of formal methods

Set based semantics (logic, algebras etc...)

Hard for software engineers to apply in practice

Lack of good software development tools based on formal approaches

Only used by well trained experts

High cost ⇔ low productivity
Outline
Aims to combine the intuition from graphical modeling languages with the semantic rigor of formal methods

Based on Generalized Sketches

Sketches are graphical representations of category theory, constraints expressed by universal constructions (limits and co-limits) of diagrams

Generalized Sketches are sketches extended with constraints as arbitrary constructions, not only universal constructions

In DPL, sketches are generalized and adapted to model driven software engineering
DPL will be used as a framework for defining modeling languages and transformations between them in a formal, graphical way.

We work continuously on Investigating, Analyzing, Adapting, Verifying and Evaluating DPL.

Potentials to use the machinery from category theory to implement MDE tools.
<table>
<thead>
<tr>
<th>Π</th>
<th>α</th>
<th>visualization</th>
<th>intended semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>[total]</td>
<td>1 → 2</td>
<td><img src="image" alt="Visualization" /></td>
<td>∀a ∈ A :</td>
</tr>
<tr>
<td>[key]</td>
<td>1 → 2</td>
<td><img src="image" alt="Visualization" /></td>
<td>∀a, a' ∈ A : a ≠ a' implies f(a) ≠ f(a')</td>
</tr>
<tr>
<td>[inv]</td>
<td>1 ↘ 2</td>
<td><img src="image" alt="Visualization" /></td>
<td>∀a ∈ A , ∀b ∈ B : b ∈ f(a) iff a ∈ g(b)</td>
</tr>
<tr>
<td>[jointly-key] or [1-1]</td>
<td>1 → 2</td>
<td><img src="image" alt="Visualization" /></td>
<td>∀a, a' ∈ A : a ≠ a' implies f(a) ≠ f(a') or g(a) ≠ g(a')</td>
</tr>
</tbody>
</table>
Diagrammatic Specification – Example

Person

Employment

Company

Employment

Person

Company

Constraints S(Pi):

[inv]

[1-1]
Instance of Diagrammatic Specification – Example

1. **Ola** from Bergen at the UiB company hires **Per** from Bergen at the UiB company as an employee on 01.01.2007. The salary is 20,000 NOK.

2. **Petra** from Bergen at the HiB company hires **UiB** from Bergen at the HiB company as an employee on 01.01.2008. The salary is 10,000 NOK.

3. **UiB** from Bergen at the UiB company hires **UiB** from Bergen at the HiB company as an employee on 01.01.2007. The salary is 30,000 NOK.

Adrian Rutle  DPL, MDE and MT (http://gs.hib.no)
## OMG Metamodel Levels

<table>
<thead>
<tr>
<th>OMG levels</th>
<th>OMG Standards/examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_3$: Meta-metamodel</td>
<td>MOF</td>
</tr>
<tr>
<td>$M_2$: Metamodel</td>
<td>UML language</td>
</tr>
<tr>
<td>$M_1$: Model</td>
<td>A UML model: Class &quot;Person&quot; with attributes &quot;name&quot; and &quot;address&quot;</td>
</tr>
<tr>
<td>$M_0$: Instance</td>
<td>An instance of &quot;Person&quot;: &quot;Ola Nordmann&quot; living in &quot;Sotraveien 1, Bergen&quot;</td>
</tr>
<tr>
<td>OMG levels</td>
<td>EMF/examples</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$M_3$: Meta-metamodel</td>
<td>Ecore</td>
</tr>
<tr>
<td>$M_2$: Metamodel</td>
<td>Ecore</td>
</tr>
<tr>
<td>$M_1$: Model</td>
<td>An Ecore model: EClass &quot;Person&quot; with EAttributes &quot;name&quot; and &quot;address&quot;</td>
</tr>
<tr>
<td>$M_0$: Instance</td>
<td>An instance of &quot;Person&quot;: &quot;Ola Nordmann&quot; living in &quot;Sotraveien 1, Bergen&quot;</td>
</tr>
</tbody>
</table>
Ecore model expressed in DPL

Adrian Rutle | DPL, MDE and MT (http://gs.hib.no)
Ecore model expressed in DPL
Ecore model expressed in DPL

EPackage

EClassifier

eClassifiers
eSuperType
disjoint
eSuperType
typedBy
eStructuralFeatures

EClass
eAttributes

eReferences

EDataType
Model Transformations

- Generation of target models from source models
- Examples:
  - Development process: Code generation, refinement etc
  - Model management: integration, decomposition etc
  - Migration: from a platform/implementation technology/programming language to another
  - Technology mappings: Java classes to Relational tables, objects to rows in database tables
Model Transformation Concepts

- Model Transformation = set of Transformation Rules + Coordinations (⇒ operational)
- Model Transaction = declarative definition of the transformation
- Transformation Engine = executes the rules
Model Transformation

MT Definition

conformsTo

MT Execution

MT Language

[Diagram of model transformation process with examples of EClass, EAttribute, EReference, DataType, Table, Column, PrimaryKey, ForeignKey, and their relationships.]

Adrian Rutle
DPL, MDE and MT (http://gs.hib.no)
Transformation Rules

\[ G(M_1) \xleftarrow{\iota_P} P \xrightarrow{m} \llbracket t \rrbracket(m) \xrightarrow{\iota_P'} P' \xrightarrow{[\cdot]} G(M_2) \]

- Input and output patterns \( P \) and \( P' \)
- Input and output instances \( I \) and \( I' \)
- The models \( M_1 \) and \( M_2 \)
- The matches \( m \) and \( \llbracket t \rrbracket(m) \)
Transformation Rules

\[ G(M_1) \leftarrow \overset{\text{trace}}{\longrightarrow} \overset{\text{trace}}{\longrightarrow} G(M_2) \]

- Input and output patterns \( P \) and \( P' \)
- Input and output instances \( I \) and \( I' \)
- The models \( M_1 \) and \( M_2 \)
- The matches \( m \) and \( \llbracket t \rrbracket (m) \)
MT Definition: Ecore to RDB
The Input EMF Model

- **eAttribute**
  - **name**: EAttribute
    - **eAttributeType**: String
      - **upperBound**: 1

- **eReference**
  - **worksFor**: EReference
    - **eReferenceType**: Person

- **eReference**
  - **hires**: EReference
    - **upperBound**: -1

- **Person**: EClass

- **Company**: EClass

Adrian Rutile | DPL, MDE and MT (http://gs.hib.no)
Rule1: EClass to Table + PrimaryKey

1:EClass

1:Table

2:Column

3:Primarykey

1:ownedby

[1]
cols

[⊂]
pcols

[1]
table

[1]
[key]
Rule 2: EAttribute to Column

1:EClass

- eAttributes

2:EAttribute

- eAttributeType

3:EDataType

1:Table

- ownedby

2:Column

- cols

3:DataType

[1]
Rule 4: 1-* EReferences to ForeignKey

1: EClass

2: EClass

3: EReference

4: EReference

5: PrimaryKey

6: Column

1: Table

3: Column

4: ForeignKey

2: Table

6: Column

Adrian Rutle
DPL, MDE and MT (http://gs.hib.no)
Rule 5: \( \star \star \) EReferences to Foreign Key

Diagram showing relationships between EClass, EReference, Column, Foreignkey, and Primarykey.
The Output RDB Model

Adrian Rutle
DPL, MDE and MT (http://gs.hib.no)
Tools implemented as plug-ins to the Eclipse platform

Need tools for:

- Drawing of models (GMF-based editor), partly finished June 2008
- Serialization of models, work started
- Validation of models, future research project
- Model transformation tools, research started
Generation of GMF-based Editors

Create GMF Project

Develop Domain Model

Develop Graphical Definition

Develop Tooling Definition

Develop Mapping Model

Create Generator Model

Generate Diagram Plug-in
The MDD is a promising approach for software development processes.
The MDD requires tools for automatic model management.
Automatic model management requires formal modeling and model transformation languages.
Using DPL as a generic framework for formalization of modeling and model transformation languages.
Lessons learned from studying the Eclipse platform.