Outline

- Metamodels are Languages – A Motivation
- Metamodeling – Examples
- MOFLON – Scenarios
- Demo (Tool Integration Scenario – TiE-DEA)
- MOFLON – Architecture
- Related Work
- Current and Future Activities at Real-Time Systems Lab
Metamodelling – Overview and Motivation

World of Computers

- Human level of abstraction
- Natural Language
  - Visual Modeling Language (e.g., UML / MOF / EMF)
  - General Purpose Language (e.g., Java, C++)
  - Bytecode
- Computer

Data in Tools complies to Datastructure, i.e. Metamodel, i.e. Language

- Tool A
  - System Requirements

- Tool B
  - System Modeling
  - Test Cases

- Tool C
  - SW-Functionality

- Tool D
  - SW-Functionality

- Metamodel Defines Language for Models
- Models representing the World

Metamodeling: Artifacts and Procedures in World (or Universe)

- Software Development
- "Hollywood"
- Astronomy
- Sports
- Traffic Management Systems
- Finance
Metamodelling – Goals

Constraints

- Constraints for detailed definition of language
- Definition of erroneous states
- Rules to comply with special design guidelines

Metamodel

- (Meta-)Modeling of language constructs
- Definition of language structure
- Domain specific semantics

- Transformationen to repair erroneous models
- Conversion of incompatible models into design compliant models
- Automatic adaption to design guidelines

Abstract Syntax

Transformation

Model
A Solution

SDM = Pattern-based Transformation Language
TGG = Bi-directional Transformation Language with TraceLinks

Abstract Syntax
MOF 2.0

Constraints
OCL 2.0

Transformation
Story Driven Modeling (SDM)
Integration
Triple Graph Grammars (TGG)

Model
MOFLON can be used to build editors, but building editors is not the main goal of MOFLON.

MOFLON is mainly used to:
- integrate existing DSL tools
- generate standard compliant metamodel implementations
- specify transformations on instances of the metamodel

Editor:
- data structure (MOFLON repository)
- GUI (GEF)
Tool Integration Example – DOORS / EA

traceability link objects
Outline

- Metamodels are Languages – A Motivation
- Metamodeling – Examples
- MOFLON – Scenarios
- Demo (Tool Integration Scenario – TiE-DEA)
- MOFLON – Architecture
- Related Work
- Current and Future Activities at Real-Time Systems Lab
Demo – Tool Integration Scenario (DOORS - EA)

DOORS Metamodel

TGGs relate

EA Metamodel

MOFLON generates

integration rule code
MOFLON – Architecture

CASE Tools
(Rational Rose, etc.)

Domain Specific Meta Models, Tool Representations

MOFLON

XML Interchange
(XMI, GXL)

Visual MOF 2.0 Editor

Visual SDM Editor
Fujaba

Visual TGG Editor

MOF 2.0 Metamodell

Constraints
(OCL, Java)

Graph Transformation
Fujaba

TGGs

XSLT Transformation
MOMoC

OCL Compiler
Dresden

Velocity Transformation
Fujaba

Java Representation
(JMI)

XML Representation
(XMI)

Tailored Interfaces

Reflective Interfaces

MOF 2.0 Instances

Event Notification

Constraint Checking

Repair Transformation

Model Analysis,
Model Transformation,
Model Integration,
etc.

Constraints
(OCL, Java)

Graph Transformation
Fujaba

TGGs

generate

refine

repair

transform

transform

transform

refine

instantiate

generate

Model Analysis,
Model Transformation,
Model Integration,
etc.

import

Model Analysis,
Model Transformation,
Model Integration,
etc.
Related Approaches

<table>
<thead>
<tr>
<th>standards</th>
<th>approaches based on graph-/model transformation</th>
<th>classic meta-CASE approaches</th>
<th>text based approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOF, OCL, QVT</td>
<td>Fujaba &amp; TGG</td>
<td>GME &amp; GReAT</td>
<td>AToM³</td>
</tr>
<tr>
<td>Abstract syntax</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Concrete syntax</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Static semantics</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Dynamic semantics</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Model analysis</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Model transformation</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Model integration</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Acceptability</td>
<td>+</td>
<td>+</td>
<td>o</td>
</tr>
<tr>
<td>Scaleability</td>
<td>+</td>
<td>+</td>
<td>--</td>
</tr>
<tr>
<td>Tool availability</td>
<td>--</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Expressiveness</td>
<td>+</td>
<td>+</td>
<td>o</td>
</tr>
</tbody>
</table>

Model-Driven Software Development at Real-Time Systems Lab

Application Areas:
Automotive SW
Automation SW

1. Meta-Models & Model Transformations (OMG)
   Sebastian Rose [EAG/SAP]
   Felix Klar [TUD/Daimler]

2. Modeling Language & Tool Integration
   Patrick Mukherjee [DFG]
   NN [DFG]

3. Domain-Specific Eng. Languages & Methods
   Elodie Legros [TUD]
   Marius Lauder [CE/Siemens]
   Anthony Anjorin [CE]
   Martin Wieber [BMBF]

   Sebastian Oster [BMBF/TUD]
   Harald Cichos [TUD]

5. Model-Driven Security Engineering
   Lars Patzina [CASED]
   Sven Patzina [TUD]

6. Model-Based & Product Line Testing

...
Further reading


Time for questions and discussion

Thank you for your attention...

http://www.moflon.org
Backup Slides
Motivation

- Models are widely used in engineering disciplines
- Need for tool support that enables model-editing
- Domain experts want domain specific languages (DSL) → domain specific models
- do not build model editors from scratch each time → reuse functionality → use meta-information
MOFLON – Main Features

- MOF2.0 editor (draw metamodels that comply to MOF2.0 standard) → build Domain Specific Languages (DSLs)
- based on the CASE-tool framework Fujaba
- possibility to extend MOFLON by own plugins
- interoperability (import / export)
- transform metamodel instances with model transformations (SDM, TGG)
- generate code (JMI-compliant) from DSLs
- instantiate models of the DSL (= repositories)
- basic editing support for generated repositories
- Standard compliance!
TiE – Architecture

Tool A

Tool B

TiE

design
(model-based)

implementation
& runtime
Tool A's data structure as metamodel

Tool A's Metamodel

Link Metamodel

Integration Rules

Tool B's Metamodel

Triple Graph Grammars relate
TiE – Architecture

MOFLON generates

Tool A

Tool B

requirements of tool integration project

MOFLON

Generated Repository

Generated Repository

Generated Integration Rule Code

Generated Repository
TiE – Architecture

implement tool adapters (handwritten code)

requirements of tool integration project

Tool A

Tool B

MOFLON

Tool Adapter A

Tool Adapter B

Generated Repository

Generated Integration Rule Code

Generated Repository

Tool Adapter Link Management
TiE – Architecture

TiE 

MOFLON

Tool A

Tool B

Tool Adapter A

Tool Adapter B

Generated Repository

Generated Repository

Integration Rule

Code

Generated Repository

Tool Adapter Link Management

requirements of tool integration project

Rule Application Strategies

Integration Framework

Editor

Visualization

controls
**TiE – Runtime**

- get element from tool
- invoke integration rule
  - find match
  - transform
  - establish link
- process next element
Meta-Metamodels – A History

- MOF 1.3/1.4
- Ecore/EMF
  - Implementation based on MOF 1.4 without Associations
- MOF 2.0
  - further development of MOF 1.4
    - from UML2.0 Constructs: Constraints, Associations, Redefinitions, Package Import, Package Merge, …
  - «merge»
  - EMOF
  - CMOF
  - Extension: Tags

~ 2000
~ 2003
2005
# Concepts in MOF 1.4, EMOF, and CMOF

<table>
<thead>
<tr>
<th></th>
<th>EMOF 2.0</th>
<th>MOF 1.4</th>
<th>CMOF 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive Type, Enumeration</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Class, Property, Operation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>- Classifier, Feature</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Packages</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>- Namespaces</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>- Inherit/Cluster/Merge</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Reference</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Association</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>- Subsets, Redefinitions</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>
# Code Generation

<table>
<thead>
<tr>
<th></th>
<th>EMOF 2.0/ Ecore</th>
<th>MOF 1.4</th>
<th>CMOF 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>EMF</td>
<td>MOIN</td>
<td>MOFLON</td>
</tr>
<tr>
<td>Interface Standard</td>
<td><em>none</em></td>
<td>JMI</td>
<td>JMI</td>
</tr>
<tr>
<td>Serialization</td>
<td>XMI</td>
<td>XMI</td>
<td>XMI</td>
</tr>
<tr>
<td>Metaobject Access</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Refl. Access on Properties</td>
<td>✓*</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Refl. Call of Methods</td>
<td>×*</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>All Instances of Classifier</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>delete() on Links/Elements</td>
<td>×*</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

* static-Method in EcoreEvaluationEnvironment/EcoreUtil