Dresden OCL2 in MOFLON

10 Jahre Dresden-OCL – Workshop

Felix Klar
Felix.Klar@es.tu-darmstadt.de

ES Real-Time Systems Lab
Prof. Dr. rer. nat. Andy Schürr
Dept. of Electrical Engineering and Information Technology
Dept. of Computer Science (adjunct Professor)
www.es.tu-darmstadt.de

© author(s) of these slides 2009 including research results of the research network ES and TU Darmstadt otherwise as specified at the respective slide
Outline

- Metamodels are Languages – A Motivation
- Metamodelling – Goals
- MOFLON – OCL
- MOFLON – Architecture
- MOFLON – Scenarios
- Demo (Integration Scenario – TiE-CDDS)
- Future Activities
Metamodeling – Overview and Motivation

(Materialized) Artifacts and Procedures in World (or Universe)
Metamodeling – Overview and Motivation

World of Computers

documents (text, model, …)

relation / dependency

Models representing the World

(Materialized)
Artifacts and Procedures in World (or Universe)

Software Development
Astronomy
“Hollywood”

Sports
Traffic Management Systems
Finance
Metamodelling – Overview and Motivation

World of Computers

(Materialized) Artifacts and Procedures in World (or Universe)

Tool A
System Requirements

Tool B
System Modeling

Tool C
SW-Functionality

Tool D
Test Cases

documents (text, model, …)
relation / dependency

Models representing the World

Software Development
„Hollywood“
Astronomy
Sports
Traffic Management Systems
Finance
Metamodelling – Overview and Motivation

World of Computers

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

Tool A
System Requirements

Tool B
System Modeling

Tool C
SW-Functionality

Tool D
Test Cases

documents (text, model, …)

relation / dependency

General Purpose Language
(e.g., Java, C++)

Bytecode

Computer

Metamodel
Defines Language for Models

Models representing the World

(Materialized)
Artifacts and Procedures in World (or Universe)

Software Development

Hollywood

Astronomy

Sports

Traffic Management Systems

Finance

a domain

Software

Development

15.10.2009
Dresden OCL2 in MOFLON
Metamodelling – Overview and Motivation

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

World of Computers

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

- Computer

Metamodel
- Defines Language for Models
- Models representing the World

Tool A
- System Requirements

Tool B
- System Modeling

Tool C
- SW-Functionality

Tool D
- Test Cases

documents (text, model, …)

relation / dependency

Software Development
- „Hollywood“

Astronomy

Sports

Traffic Management Systems

Finance

(Materialized)
Artifacts and Procedures in World (or Universe)
Metamodeling – Goals

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

- (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

Metamodel

Model

Transformation
A Solution

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

MOF 2.0

Abstract Syntax

Constraints

OCL 2.0 (Dresden OCL)

Transformation

Story Driven Modeling (SDM)

Integration

Triple Graph Grammars (TGG)

Model
(OCL) Constraints in MOFLON – MOF Editor

- MOF allows to add constraints to every MOF element
- MOFLON has an underlying MOF metamodel repository
  → MOFLON MOF editor may add constraints to elements
MOFLON allows to generate metamodel implementations (Java/JMI)

- MOFLON uses Dresden OCL to add constraint code to generated implementation
  - invariants (inv)
  - derived attributes (derive)
  - helper variables/functions (def)

**Dresden OCL-code**

refVerifyConstraint(String name):JmiException

**MOFLON-code**

refVerifyConstraints(boolean deepVerify):Collection

**JMI compliant method**

refVerifyConstraints(boolean deepVerify):Collection

**generated Repository**

Model A

c1:Clazz

**<<invokes>>**

**<<calls>>**

**<<queries>>**
**MOFLON – Architecture**

MOFLON

- Domain Specific Meta Models, Tool Representations
- **CASE Tools** (Rational Rose, etc.)
- XML Interchange (XMI, GXL)
- Visual MOF 2.0 Editor
- Visual SDM Editor (Fujaba)
- Visual TGG Editor
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

import

instantiate
instantiate
instantiate
instantiate
instantiate

refine
refine
repair

generate
MOFLON – Architecture

MOFLON

XML Interchange (XMI, GXL)  Visual MOF 2.0 Editor  Visual SDM Editor  Visual TGG Editor

CASE Tools (Rational Rose, etc.)  import

MOF 2.0 Metamodel

Constraints (OCL, Java)

Graph Transformation  Fujaba

TGGs

XSLT Transformation  MOMoC

OCL Compiler  Dresden

Velocity Transformation  Fujaba

Domian Specific Meta Models, Tool Representations
Case Study – Statechart Editor (STaX)

Editor:
- data structure (MOFLON repository)
- GUI (GEF)

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
Integration Example –
Class diagrams / database schemata

domain specific language, e.g. Class Diagrams

domain specific language, e.g. Database Schemata
Tool Integration Scenario (CD / DS)

Class Diagrams Metamodel

TGGs relate

Database Schemata Metamodel

Run-Time Verification of Constraints

MOFLON generates

integration rule code
Outline

- Metamodels are Languages – A Motivation
- Metamodeling – Goals
- MOFLON – OCL
- MOFLON – Architecture
- MOFLON – Scenarios
- Demo (Integration Scenario – TiE-CDDS)
- Future Activities
TiE-CDDS – Focus on Constraints in CD (1)
Generate Code from MOF model (CD metamodel)
TiE-CDDS – Focus on Constraints in CD (2)
Integration Framework

- load CD metamodel
- load CD model

Visualization of class diagrams model (here: source domain)

Model violates constraints:
- Class "Customer" has two attributes with same name: "name"
- Attribute in class "Address" has no name
- Multiplicity violation: class "Order" has no attribute but according to CD metamodel every class must have one
TiE-CDDS – Focus on Constraints in CD (3)
Model Browser

model is fixed in generic model editor
TiE-CDDS – Focus on Constraints in CD (4)
Integration Framework

Translation process may start now...

source domain model fulfills its constraints
TiE-CDDS – Focus on Constraints in CD (5)
Forward Translation to DB representation
Model-Driven Software Development at Real-Time Systems Lab

Application Areas:
Automotive SW
Automation SW
...

1. Meta-Models & Model Transformations (OMG)
2. Modeling Language & Tool Integration
3. Domain-Specific Eng. Languages & Methods
5. Model-Driven Security Engineering
6. Model-Based & Product Line Testing

Patrick Mukherjee [DFG]
NN [DFG]

Sebastian Rose [EAG/SAP]
Felix Klar [TUD/Daimler]

Elodie Legros [TUD]
Marius Lauder [CE/Siemens]
Anthony Anjorin [CE]
Martin Wieber [BMBF]

Sebastian Oster [BMBF/TUD]
Harald Cichos [TUD]
Lars Patzina [CASED]
Sven Patzina [TUD]

5. Model-Driven Security Engineering
6. Model-Based & Product Line Testing
Future Work – OCL

- Activate more features of Dresden OCL in MOFLON
  - MOF editor
    - User friendly OCL syntax checking
    - OCL expression completion
  - MOFLON code generator
    - Initial Values (init)
  - Queries?
  - ...

- We bootstrap our MOFLON MOF Metamodel periodically
  - Add more OCL constraints to our MOF Metamodel
  - Regenerate MOFLON MOF implementation
  - Activate constraint checking in MOFLON
    → Model Verification
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!
## 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, Fujaba &amp; TGG, PROGRES &amp; TGG, EMF &amp; Teikat, AToM³, MetaEdit+, EBNF &amp; TXL, SQL, XML</td>
<td></td>
<td></td>
<td></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>+</td>
</tr>
<tr>
<td>Scaleability</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Tool availability</td>
<td>--</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Expressiveness</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>