6.3.2 Metamodeling in MetaCASE MOFLON

From: 10 Jahre Dresden-OCL – Workshop
http://dresden-ocl.sourceforge.net/
http://dresden-ocl.sourceforge.net/10years.html
Outline

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

(Metazized) Artifacts and Procedures in Worlds

Hollywood

Software Development

Astronomy

Sports

Traffic Management Systems

Finance

a domain
Metamodeling – Overview and Motivation

World of Computers

Models representing the World

documents (text, model, …)

relation / dependency

Software Development
Astronomy
Hollywood
Finance
Traffic Management Systems
Sports

(Metamodelized)

MemPrO Procedures

(Universe)
Metamodelling – Overview and Motivation

World of Computers

1. Tool A: System Requirements
2. Tool B: System Modeling
3. Tool C: SW-Functionality
4. Tool D: Test Cases

Models representing the World

Documents (text, model, …)

Relation / dependency

Domain: System of Systems

Software Development

Domain: Software

Astronomy

Domain: Science and Research

Sports

Domain: Entertainment

Traffic Management Systems

Domain: Industry

Finance

Domain: Economy

Metamodelling (Materialised) Artifacts and Procedures (Universe)

Hollywood™
Metamodeling – Overview and Motivation

World of Computers

- General Purpose Language (e.g., Java, C++)
- Bytecode
- Metamodel
  - Defines Language for Models
- Models representing the World

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

- "Hollywood"
  - (Materialized) Procedures (Universe)
- Software Development
- Astronomy
- Sports
- Traffic Management Systems
- Finance

A domain

15.10.2009 | Dresden OCL2 in MOFLON
Metamodelling – Overview and Motivation

World of Computers

<table>
<thead>
<tr>
<th>level of abstraction</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Natural Language</td>
</tr>
<tr>
<td></td>
<td>Visual Modeling Language (e.g., UML / MOF / EMF)</td>
</tr>
<tr>
<td></td>
<td>General Purpose Language (e.g., Java, C++)</td>
</tr>
<tr>
<td></td>
<td>Bytecode</td>
</tr>
</tbody>
</table>

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

Tool A
System Requirements

Tool B
System Modeling

Tool C
Test Cases

Tool D
SW-Functionality

Field of Application

- Software Development
- (Materialized) Procedures (Universe)
- Software
- Astronomy
- Sports
- Traffic Management Systems
- Finance

Metamodel
Defines Language for Models

Models representing the World

"Hollywood"

<<instanceof>>
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
Metamodeling – 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

Transformation

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

Abstract Syntax

Model
A Solution

MOF LON meta-CASE tool

Constraints
OCL 2.0 (Dresden OCL)

Abstract Syntax
MOF 2.0

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

Model

SDM = Pattern-based Transformation Language
TGG = Bi-directional Transformation Language with TraceLinks
MOFLON MetaCASE – 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!
Einschub: JMI: Transformative TS-Brücke für MOF und Java, Sprache UML

Java Metadata Interchange (JMI) ist eine TS-Halb-Brücke für MOF und EBNF-Space, für die Sprache UML
(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

validate constraints
(OCL) Constraints in MOFLON – Generated Implementations

- MOFLON generates metamodel-based repositories (Java/JMI)
- MOFLON uses Dresden OCL to add constraint code to generated implementation
  - invariants (inv)
  - derived attributes (derive)
  - helper variables/functions (def)

MOFLON-code
refVerifyConstraint(String name):JmiException

JMI compliant method
refVerifyConstraints(boolean deepVerify):Collection

Dresden OCL-code
- generated Repository
  - c1:Clazz

Model A
public Collection<String> refConstraintNames() {
    Collection<String> constraintNames = new java.util.HashSet<String>();

    constraintNames.add("attrNamesMustDiffer");

    return constraintNames;
}

public javax.xml.reflect.JmiException refVerifyConstraint(String constraintName) {
    if ("attrNamesMustDiffer".equals(constraintName)) {
        if (!evaluate_attrNamesMustDiffer()) {
            String constraintBody = "inv:attrs->forall(a1,a2:Attribute|a1<>a2 implies a1.name <> a2.name)";
            informListener(new ConstraintEvent(this, ConstraintEvent.EVENT_OCL_INVARIANT, "constraintName", false));

            return new javax.xml.reflect.ConstraintViolationException(
                constraintBody, this, "constraint named " + constraintName + " is violated in instance: " + this);
        }
        else {
            informListener(new ConstraintEvent(this, ConstraintEvent.EVENT_OCL_INVARIANT, "constraintName", true));
        }
    }
    return null;
}

public Collection<javax.xml.reflect.JmiException> refVerifyConstraints(boolean deepVerify) {
    Collection<javax.xml.reflect.JmiException> invalidConstraints = new org.moflon.collections.implementation.JmiSetImpl<>

    for (String constraintName : refConstraintNames()) {
        javax.xml.reflect.JmiException constraintException = refVerifyConstraint(constraintName);

        if (constraintException != null) {
            invalidConstraints.add(constraintException);
        }
    }

    if (deepVerify) {
    }

    if (invalidConstraints.size() > 0) {
        return invalidConstraints;
    } else {
        return null;
    }
}
```java
// generating constraint evaluation method attrNamesMustDiffer

public boolean evaluate(attrNamesMustDiffer) {

    // Variables
    final tudresen.ocl20.core.lib.OclCollectionType tudocl20Type1 = tudresen.ocl20.core.lib.OclCollectionType.getOclCollectionType("cd_metamodel::Attribute").getOclCollectionType();
    final tudresen.ocl20.core.lib.OclModelType tudocl20Type0 = tudocl20Type0.getOclModelTypeFor("cd_metamodel::OclClass");

    // Invariant
    final tudresen.ocl20.core.lib.OclModelObject tudocl20Var0 = (tudresen.ocl20.core.lib.OclModelObject) tudocl20Fact0.getOclRepresentationFor(tudocl20Type0, this);
    final tudresen.ocl20.core.lib.OclBag tudocl20Exp0 = tudresen.ocl20.core.lib.OclBag.tocl20Bag(tudocl20Var0, tudocl20Type0, "attr");
    final tudresen.ocl20.core.lib.OclIterator tudocl20Iter0 = tudocl20Exp0.getIterator();
    final tudresen.ocl20.core.lib.OclBooleanEvaluable tudocl20Eval0 = new tudresen.ocl20.core.lib.OclBooleanEvaluable() {
        public tudresen.ocl20.core.lib.OclBoolean evaluate() {
            final tudresen.ocl20.core.lib.OclModelObject tudocl20Var1 = tudresen.ocl20.core.lib.OclModelObject.tocl20ModelObject(tudocl20Iter0, tudocl20Var0, tudocl20Type0);
            final tudresen.ocl20.core.lib.OclIterator tudocl20Iter1 = tudocl20Var1.getIterator();
            final tudresen.ocl20.core.lib.OclBooleanEvaluable tudocl20Eval1 = new tudresen.ocl20.core.lib.OclBooleanEvaluable() {
                public tudresen.ocl20.core.lib.OclBoolean evaluate() {
                    final tudresen.ocl20.core.lib.OclModelObject tudocl20Var2 = tudresen.ocl20.core.lib.OclModelObject.tocl20ModelObject(tudocl20Iter0, tudocl20Var0, tudocl20Type0, tudocl20Type1, tudocl20Type0, tudocl20Type1);
                    return tudocl20Var1.isNotEqual(tudocl20Var2);
                }
            }
            return tudocl20Exp0.isNotEqual(tudocl20Eval0);
        }
    };

    return tudocl20Exp6;
};

final tudresen.ocl20.core.lib.OclBoolean tudocl20Exp6 = (tudresen.ocl20.core.lib.OclBoolean) tudocl20Exp0.forAll(tudocl20Iter1, tudocl20Eval1);

return tudocl20Exp7.isTrue();
}
```
XML Interchange (XMI, GXL) | Visual MOF 2.0 Editor | Visual SDM Editor | Visual TGG Editor

CASE Tools (Rational Rose, etc.)

Domain Specific Meta Models, Tool Representations
MOFLON – Architecture

MOFLON

CASE Tools
(Rational Rose, etc.)

import

Domain Specific Meta Models, Tool Representations

MOF 2.0 Metamodel

Constraints
(OCL, Java)

Graph Transformation
Fujaba

TGGs

generate

refine
repair

instantiate

XML Interchange
(XMI, GXL)

Visual MOF 2.0 Editor

Visual SDM Editor
Fujaba

Visual TGG Editor

CASE Tools
(Rational Rose, etc.)

import
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

instantiate
instantiate
instantiate
instantiate

refine
refine
repair

transform
transform
transform

MOF 2.0 Metamodell
Constraints (OCL, Java)
Graph Transformation Fujaba
TGGs

generate

XSLT Transformation MOMoC
OCL Compiler Dresden
Velocity Transformation Fujaba

MOFLON – Architecture

CASE Tools (Rational Rose, etc.)
MOFLON – Architecture

CASE Tools
(Rational Rose, etc.)

Model Analysis,
Model Transformation,
Model Integration,

Domain Specific Meta Models, Tool Representations

MOFLON

XML Interchange
(XMI, GXL)

Visual MOF 2.0 Editor
Fujaba

Visual SDM Editor

Visual TGG Editor

instantiate

instantiate

instantiate

refine

refine

transform

transform

transform

generate

generate

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.
Case Study – Statechart Editor (STaX)

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)
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

MOFLON generates

integration rule code

Run-Time Verification of Constraints
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)

**Constraint Validation**

source domain model does not fulfill its 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

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…

Constraint Validation

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 (Prof. Schürr)

6. Model-Based & Product Line Testing


5. Model-Driven Security Engineering

2. Modeling Language & Tool Integration

3. Domain-Specific Eng. Languages & Methods

1. Meta-Models & Model Transformations (OMG)

Application Areas:
Automotive SW
Automation SW
...

15.10.2009 | Dresden OCL2 in MOFLON
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
## 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>Tefkat</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>0</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>0</td>
</tr>
<tr>
<td>Scalability</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>0</td>
</tr>
</tbody>
</table>

Further reading


Time for questions and discussion

Thank you for your attention...

http://www.moﬂon.org