From Model-Driven Development to Graph Transformations and Back Again

A. Schürr
Real-Time Systems Lab
TU Darmstadt
andy_schuerr@es.tu-darmstadt.de

SegraVis Advanced Summer School
8th – 11th September 2006
Leicester, UK

OR:

„Smuggling“ (Fujaba’s) Graph Transformations into the World of OMG Standards
Outline of Presentation

1. Languages and Tools for Model-Driven Development
   - OMG’s Model Driven Architecture (MDA)
   - Model-Driven Software Development (MDD)
   - MDD requirements derived from industrial case study

2. From MDD to the World of Graph Transformations
   - Comparison of Meta-Case, Model/Graph Transformation Tools
   - MOFLON = OMG standards + graph transformation technology
   - MOFLON architecture and sublanguages

3. … and Back Again
   - Status quo and future of MOFLON
   - Status quo of MDA/MDD/DSL/Meta-Case/… tools in general
**OMG’s Model Driven Architecture**

OMG often recommends:

- DSL1 = DSL2 = DSL3 = UML

- CIM = Computer Independent Model
- PIM = Platform Independent Model
- PSM = Platform Specific Model
- MOF = Meta Object Facility (NOT Microsoft Ops. Framework)
- UML = Unified Modeling Language
- DSL = Domain-Specific Language

**Modeling Levels**

1. **M0**: Infrastructure = UML ∩ MOF
2. **M1**: System
3. **M2**: UML Metamodel
4. **M3**: DSL Metamodel

The diagram illustrates the relationship between different modeling levels and metamodels, highlighting the transformation and mapping between Computational Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM), ultimately leading to Code and System.
Model-Driven Development Requirements

Rapid Development (generating) of

- Tool wrappers for „COTS“ tools
- New tools for domain-specific languages
- Local model analysis/transformation support
- Inter-model consistency checking / change propagation
- Traceability link management support
- Bidirectional model update propagation
- Data import / export (code generators, parser, …)
- Integrated model version management
- …
SEGRAVIS-Leicester, 2006-09-09

www.es.tu-darmstadt.de

Fachgebiet Echtzeitsysteme

Model-Driven Development Architecture

- Representation Grammar
- Meta Model
- Analysis Results
- Model Constraints
- Model Transformations
- GUI Specification
- Model Translations

Model Parser & Generator
(PIM) Meta Objects
Model Translator
(PSM) Meta Objects

Code Parser & Generator

Text Templates

Model Representation

Text<>Model Translations

Code Fragments

Representation Grammar

Meta Model

Analysis Results

Model Analyser
Model Transformer
Model Editor

Model Constraints
Model Transformations
GUI Specification

Model Translations

Model Editor
Model Transformer

Meta-Case Tools usually generate Domain-Specific Model Editors (+ Analyzer + … in some cases)

Meta Model

Analysis Results

Model Analyser
Model Transformer
Model Editor

GUI Specification

Model Constraints
Model Transformations

Model Translations

Model Editor
Model Transformer

Text Templates

Model Representation

Code Fragments

Standard Meta-Case Tool (DSL) Scenario
Elements of a Complete Meta-DSL

- Code/Text Generator & Parser
- GUI
- Metamodel
- Constraint
- Transformation
- Model-to-Model Translation

Elements of OMG’s MDA World

- MOF 2.0
- OCL 2.0
- QVT 1.0
- No GUI Definition Language
- Model-to-Text Transformation RFP
- M2M Translation Def. Language
The Meta-Case Tool World of MetaEdit+

Elements of Fujaba’s Modeling World
### MDD / DSL Languages - Summary

<table>
<thead>
<tr>
<th></th>
<th>OMG Languages</th>
<th>AMMA (INRIA)</th>
<th>Arc-Styler</th>
<th>GME (Vanderbilt)</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metamodel Def. Lang.</td>
<td>MOF</td>
<td>KM3</td>
<td>UML Profile</td>
<td>GME 5.0</td>
<td>UML 1.x</td>
</tr>
<tr>
<td>GUI Def. Lang.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GME 5.0</td>
<td>-</td>
</tr>
<tr>
<td>Constraint Def. Lang.</td>
<td>OCL</td>
<td>ATL / OCL</td>
<td>OCL</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Model Trafo Def. Lang.</td>
<td>QVT</td>
<td>ATL</td>
<td>?</td>
<td>GReAT</td>
<td>SDM</td>
</tr>
<tr>
<td>M2M Trans. Def. Lang.</td>
<td>QVT</td>
<td>ATL / AMW²</td>
<td>AIM²</td>
<td>GReAT²</td>
<td>TGG</td>
</tr>
<tr>
<td>Code Gen. Def. Lang.</td>
<td>-</td>
<td>TCS</td>
<td>Cart-ridges</td>
<td>-</td>
<td>Velocity</td>
</tr>
</tbody>
</table>

1: QVT has been designed for model-to-model translation purposes
2: ATL, AIM, and GReAT are unidirectional model translation languages

### MOFLON = OMG Standards + Fujaba

- **MOF 2.0** as metamodeling language
  - standard-complaint metamodeling approach
  - new modularization / model refinement concepts

- **OCL 2.0** as constraint definition language
  - well-known textual model property definition language
  - Dresden OCL Compiler (planned add-on: incr. evaluation)

- **SDM = graph transformations + UML activity diagrams**
  - UML-inspired visual model transformation language
  - multi-paradigmatic (rule-based, imperative) approach

- **(V)TGGs = MOF-compliant triple graph grammars**
  - bidirectional, declarative model translation approach
  - planned add-on: declarative view-definition approach

- **JMI-compliant Java code generators (Sun Standard)**
With further 86 constraints 101 improvements were made, whereas

From 90 analyzed constraints of the specification
42 (47%) are erroneous
48 (53%) are correct

With further 86 constraints 101 improvements were made, whereas
50 (50%) are additional
51 (50%) are corrections

From the 51 corrections of the specification
16 (29%) are due to a wrong metamodel reference
19 (34%) are due to wrong semantics
21 (37%) are due to wrong syntax
Typical MOFLON Applications

- system engineering **tool integration**
  (ToolNet project with DaimlerChrysler et al.)
- model analysis / **design guideline checking**
  (Matlab Simulink/Stateflow with DaimlerChrysler et al.)
- software analysis / **reverse engineering**
  (based on experiences at Philips Medical Research)
- visual **DSL editor development**
  (ECLIPSE plug-ins in cooperation with UniBw)
- ...

MOFLON Architecture

- Meta Models
- CASE Tools
- Import
- XMI
- MOF 2.0
- OCL
- Visual Editors
- Code Generators
- JMI
- Access methods (typed, reflective)
- XML persistency
- Extra Features
- Event Notification
- Constraint checking & repair
- Tool Logic
Running Example: Model integration

Model Instance = Object Graph

:FormalModule
name="Cruise Control"

:FormalObject
heading="UseCase"

:FormalObject
heading="Detect ..."

:FormalObject
heading="Attributes"

:FormalObject
heading="Control..."
Metamodel for DOORS

Metamodel for Enterprise Architect
Triple Graph Grammar Integration Schema

Graph Schema 1

- EAPackages
- EAConnectors
- EElements
- FormalObject
- FM_P
- FO_P

Graph Schema 2

Graph Schema 3

Triple Graph Grammar Integration Rule

Object Graph 1

Object Graph 2

Object Graph 3
Generating Operational Rules

- Integration rules are declarative
- Standard SDM rules are generated:
  - check consistency
  - create traceability links
  - forward transformation
  - backward transformation
  - forward attribute propagation
  - backward attribute propagation
  - remove traceability links
  - forward deletion propagation
  - backward deletion propagation
  - ...

Example: Generate use case diagrams from requirements
Java Code Generation from SDM

- JMI-compliant Java code
  - in-memory repository
  - type-safe interfaces
  - reflective/generic interfaces
  - XML persistency
- operational transformation rules → method bodies
- MDR-compatible interfaces and event mechanism (MDR = Sun’s Meta Data Repository)
- Under Development:
  - code for constraint checking (from OCL constraints)
  - code for constraint repairs (from transformations)
Example: Developed Tool Integration

Generic CASE Tool Integrator Framework (ToolNet)
(with Rule Application Strategy)

Source Model
Integration Model
Target Model

Reusable
DOORS / JMI Adapter

Generated
Integration Rules

Reusable
EA / JMI Adapter

Proprietary Model

Traceability Link Repository (RDBMS)

Tool Adapter Generation Strategies

Typed JMI Interface
Generated JMI / JMI Adapter

Refl./Generic JMI Interface

COTS Tool API
COTS Tool Impl.

Extensible Adapter Realization

Typed JMI Interface
Generated Model Repository
Refl./Generic JMI Interface

Manual Ex-/Import Tool Impl.

COTS Tool API
COTS Tool Impl.

Export / Import Batch Realization

Typed JMI Interface

Refl./Generic JMI Interface
Generated JMI / JMI Adapter

Manually JMI / Tool Adapter Impl.

COTS Tool API
COTS Tool Impl.

Hard-Wired Adapter Realization

Manual JMI / Tool Impl.

COTS Tool API
COTS Tool Impl.
Open Problems & Ongoing Work

- Metamodeling with MOF 2.0
  - missing UML concepts (association classes)
  - integration with UML profile definition

- Constraint Definition with OCL 2.0
  - incremental (event-driven) constraint checking
  - integration with transactions & repair actions

- Local Model Transformations with SDM
  - handling of composition hierarchies (still a problem!)
  - integrated formal definition of language mix

- Model To Model Transformations with TGGs
  - handling of complex attribute evaluations (still a problem!)
  - ...

- Integration with Editor Generator Framework DiaGen
• Model-Driven Development (MDD) is a “hot topic” of the Software Engineering Community
  ➢ with all the resulting pros and cons …

• MDD combines established techniques of last millenium
  ➢ meta-modeling / meta-case tool technology
  ➢ stepwise refinement of specifications / models
  ➢ compiler compiler technology
  ➢ …

• OMG’s MDA and other institution’s Meta-Case Tools for Domain-Specific Languages (DSLs) are variations of MDD

• Currently available (commercial / academic) MDD tools
  ➢ support only subsets of all MDD activities
  ➢ lacks formal definition (available for graph transformations)