Model Transformation and Weaving Tools in the AMMA Platform

Jean Bézivin, Marcos Didonet Del Fabro, Frédéric Jouault
{jean.bezivin, marcos.didonet-del-fabro, frederic.jouault}@univ-nantes.fr
ATLAS group (INRIA & LINA), University of Nantes, France
http://www.sciences.univ-nantes.fr/lina/atl/

Contents

• Introduction
  • Model Engineering Overview
  • AMMA Platform Overview
• Model Transformation with ATL
• Model Weaving with AMW
• Case Study: Bridging UML Profiles and Domain Specific Languages
• Demonstrations

Model Engineering Overview

Metamodel

Classical

representation

Model

conformsTo

repOf

M1

M2

M3

Metametamodel

conformsTo

meta

AMMA Platform Overview

AMMA: a Lightweight Model Engineering Architectural Style

Model Transformation and Model Weaving

Model Management

Model Projectors

(Megamodel)
Model Transformation and Weaving Tools in the AMMA Platform

© 2005 ATLAS Nantes

Contents

- Introduction
  > Model Transformation with ATL
    - Definitions
    - ATL Overview
    - ATL Development Tools
  - Model Weaving with AMW
  - Case Study: Bridging UML Profiles and Domain Specific Languages
  - Demonstrations

Model transformation definition

- By definition, a model transformation is the automatic creation of target models from source models.
- Model transformation is not only about M1 to M1 transformations:
  - M1 to M2: promotion,
  - M2 to M1: demotion,
  - M3 to M1, M3 to M2, etc.

Some definitions

- MOF
- MMa is the source metamodel
- MBb is the target metamodel
- Ma is the source model
- Mb is the target model

Small theory

- Metametamodel
- Rule
- Rule
- MM2Mb.atl
- conformsTo
- conformsTo
- Node
- Node
- Node
- Node
- Node
- Node
- Node
- Node

- MMa
- ATL
- Mb
- Mb

- MBb
- Node
- Node
- Node
- Node
- Rule
- Rule

- Ma
- Ma
- Mb
- Mb
ATL Overview

- Source models and target models are distinct.
- Source models cannot be modified, only navigated.
- Target models cannot be navigated.
- The language is a declarative-imperative hybrid:
  - There are declarative matched rules.
  - There are imperative called rules (to be implemented).
- An imperative rule is basically a procedure.
- A declarative rule specifies:
  - A source pattern to be matched in the source models.
  - A target pattern to be created in the target models for each match.

Execution order

- The order in which rules are matched and applied is not specified.
- The order in which bindings are applied is not specified.
- The execution of declarative rules can however be kept deterministic:
  - The execution of a rule cannot change source models: it cannot change a match.
  - Target elements are not navigable: the execution of a binding cannot change the value of another.

The ATL Virtual Machine Architecture

The simplest form of a model repository is a file system.

ATL Development Tools (ADT): an Eclipse-based IDE for ATL

- ATL perspective and ATL project
- ATL Code editor
  - Syntax highlighting
  - Outline
  - Error reporting
- ATL Builder
- ATL Debugger
  - Source-level debugging
  - Stepping through elementary operations
  - Breakpoints support
Model Transformation and Weaving Tools in the AMMA Platform

ATL Development Tools: perspective, editor and outline

ATL Development Tools: launch configuration

Contents

• Introduction
• Model Transformation with ATL
  • Definitions
  • Weaving metamodel
  • Weaving as models: advantages
  • Combining model weaving and transformation
• Case Study: Bridging UML Profiles and Domain Specific Languages
• Demonstrations
Model Weaving

- Model Weaving means:
  - Establishing typed links between model elements
  - Representing these links as models
  - Defining link types in a metamodel

Model Weaving Use Cases

- Data exchange
- Schema evolution
- Heterogeneous data integration
- Mapping between models and a graphical representation
- Transformation specifications
- Etc.

Models

- Models: directed graphs
- Mappings: how to represent them?
Weaving core metamodel

• Several weaving metamodels (extending a minimal metamodel)
• we cannot predict all weaving semantics not a general purpose weaving language

Weaving as models: advantages

• Use of same model management primitives
  • Add, delete, update, navigation

• Expressiveness
  • Reason about mappings (infer about mappings)
    ("The fundamental role of a reasoning engine is to derive new information via automated inference" [sigmod04])
  • Foreign keys, nested constraints, ordering, concatenation, merge, etc.
    • Hard to express with 1-1 correspondences (semantics implemented ad-hoc)

Efficiency and reuse: weaving model + transformations

\[ \begin{align*}
  &MM_a \quad \stackrel{G}{\rightarrow} \quad MM_b \\
  &\text{Semi-automatic} \quad \text{Weaving}
\end{align*} \]

\[ \begin{align*}
  &Ma \quad \stackrel{G}{\rightarrow} \quad Mb \\
  &\text{Automatic} \quad \text{Producing T from Ma} \\
  &\text{(based on patterns)} \quad \text{Transforming models}
\end{align*} \]
Combining Model Weaving and Model Transformation

For a given weaving model, two representations: ATL and XSLT

AMW Example: represented semantics

AMW Example: some UI facilities
Two major approaches appear in Model Engineering to model domain specific concepts:
- UML approach: uses a single, large, general purpose metamodel, extensible through profiles.
- DSL (Domain Specific Languages) approach: uses a large number of small, domain specific, metamodels.

We need interoperability between these approaches:
- A tool which can perform automatic transformations between profiled UML models and models conforming to corresponding metamodels.

*Work realized by Anas Abouzahra.

UML Profiles (1/2)

- UML profiles are dedicated to the strategic intention to formalize and support the development of applications with UML:
  - It allows designers to customize UML to their particular domain or purpose.
  - It defines virtual UML subclasses by associating stereotypes, tag definitions and constraints to provide an additional meaning to UML classes.

- Profiles are generally represented by mapping tables which show correspondences between stereotypes and concepts.

Mapping table:

<table>
<thead>
<tr>
<th>MetaModel Element</th>
<th>UML Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetaModel_Class</td>
<td>The UML class and optional stereotypes, which combine to represent the MetaModel_Class</td>
</tr>
</tbody>
</table>

UML Profiles (2/2)

- Each element of the profile maps to a specific metamodel element.
- The mapping table is completed by mapping details:
  - They define more precisely each map between UML and metamodel elements.
  - Each mapping details contains subsections covering these topics: tags, mapping properties, constraints and limitations.
  - Tags are used for metamodel properties not directly supported by UML.
  - Some domain specific details cannot be rendered in UML using profiles. These details are described in a limitations section.
Example: UML Profile For MOF (1/2)

- Purpose: using widely available UML CASE tools to design metamodels
- Each element of the profile maps to a specific MOF element.

<table>
<thead>
<tr>
<th>UML Profile For MOF: concepts mapping table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metamodel Element</td>
</tr>
<tr>
<td>Package</td>
</tr>
<tr>
<td>Interaction</td>
</tr>
<tr>
<td>Operation</td>
</tr>
<tr>
<td>Constraint</td>
</tr>
<tr>
<td>Component</td>
</tr>
<tr>
<td>Node</td>
</tr>
<tr>
<td>Association</td>
</tr>
<tr>
<td>Model or Package</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>Attribute</td>
</tr>
</tbody>
</table>

Example: UML Profile For MOF (2/2)

- Each element of the profile maps to a specific MOF element as shown in the mapping table above.
- The mapping table is completed by mapping details.

Profile metamodel

- The definition of each virtual UML subclass which corresponds to a definition of a Profile element (in particular a class of the metamodel profiled) can be done in several ways.
- An UML element can take more than one stereotype to define the same element of the metamodel profiled. However, it can also take different stereotypes to define different elements.
- We should have the possibility to navigate the UML elements used by the Profile and have access to some of their attributes and values.
Implementation Architecture

Using ATL transformations (1/3)

Using ATL transformations (2/3)
Model Transformation and Weaving Tools in the AMMA Platform

Using ATL transformation (3/3)

\[ (M_{\text{uml}}, M'_{\text{uml}}) = T_{\text{uml}} \circ \text{weave}(M_{\text{MM}}, M_{\text{MAMW}}, M_{\text{profile}}, M_{\text{MAMW}}, M_{\text{MM}}) \]
\[ M_{\text{uml}} = T_{\text{uml}}(M_{\text{MM}}) \]
\[ M_{\text{MAMW}} = T_{\text{MAMW}}(M_{\text{MM}}) \]
\[ (T_{\text{uml}} \circ \text{weave})(M_{\text{MM}}) = T_{\text{MAMW}}(M_{\text{MM}}) \]

The ProfileForMOF Model

- profile ProfileForMOF : MOF {
  ...
  define AssociationEnd {
    UML!AssociationEnd {
      tag documentation ;
      -- AssociationEnd.annotation
      nested {
        ref name;
        ref namespace;
        ref constraint;
        ref participant;
        ref multiplicity {range { lower , upper }};
        ref ordering : literal {ok_ordered (true) , ok_unordered(false)} ;
        -- isOrdered is mapped to ordering (where true corresponds to ordered)
        ref technique : boolean = true;
        ref aggregation : literal {ak_composite , ak_aggregate ,ak_none};
        ref isNavigable;
        ref changeability : literal {ck_changeagle (true), ck_frozen(false) , ck_addOnly(false)};
        refl: Unique: boolean = true;
        ref aggregation : literal {ak_composite , ak_aggregate ,ak_none};
      }
    ...
  }
  ...
}

Contents

- Introduction
- Model Transformation with ATL
- Model Weaving with AMW
- Case Study: Bridging UML Profiles and Domain
- Demonstrations
End of the presentation

- Thanks
- Questions?
- Comments?