Formal Model Transformations in Model Driven Architecture

Alfonso Pierantonio
Software Engineering and Architecture Group
Università degli Studi dell'Aquila
Dipartimento di Informatica
I-67100 L'Aquila
alfonso@di.univaq.it

joint work with Davide Di Ruscio
Roadmap

» Introduction

» What is a Model?

» MDA Primer

» Model Transformations

» Example: Developing data-intensive Web Applications

» Conclusions
Introduction

» Model transformations are increasingly gaining attention in different areas of software design and integration

» Model transformation presents intrinsic difficulties

» It requires specialized support in several aspects in order to realize the full potential, for both the enduser and transformation developer [Tratt 04]

» Different proposals have been issued, especially in combination with the QVT RFP [OMG 02]

» Abstract State Machines as a candidate for specifying (and executing) model transformations
What is a Model?

In his work in the seventies H.Stachowiak characterized a model as follows

1. A model has a **purpose**

2. A model describes some **entity** that exists or is intended to exist in the future

Allgemeine Modelltheorie
Herbert Stachowiak
Springer (1973)
What is a Model?

In his work in the seventies H. Stachowiak characterized a model as follows

1. A model has a **purpose**

2. A model describes some **entity** that exists or is intended to exist in the future

3. A model is an **abstraction**, that is, it does not describe details of the entity that are not of interest to the audience of the model

*Allgemeine Modelltheorie*  
*Herbert Stachowiak*  
*Springer* (1973)
Pros

- Models help us understand a complex problem and its potential solutions through abstraction

- Characteristics
  - abstract, understandable, accurate, predictive, inexpensive

- A number of pragmatic qualities
  - improved communication of ideas
  - completeness checks
  - viability in terms of indicators such as cost and estimation
  - test case generation

Cons

- Models when used only as documentation, have a limited value since they easily diverge from reality
Models | Pros & Cons

» Pros

> Models help us understand a complex problem and its potential solutions through abstraction

> Characteristics

> abstract, understandable, accurate, predictive, inexpensive

> A number of pragmatic qualities

> improved communication of ideas
> completeness checks
> viability in terms of indicators such as cost and estimation
> test case generation

» Cons

> Models when used only as documentation, have a limited value since they easily diverge from reality
» Too many platforms and technologies
  > Distributed Objects, Components, Web services, etc
  > Which technology is the best?

» Too fast evolution
  > Technologies evolve and get obsolete very soon
  > Which technology will be out tomorrow?

» How to protect my investment in business logic?
  > The business logic has to be as independent as possible from supporting technologies
MDA | Introduction

» Defined by OMG (2000) and based on modeling and automated mappings of models to implementations

» The artifacts are formal models, i.e., models that can be understood by computers

» It separates the specification of system functionality from the specification of the implementation on a given technology platform

» **Slogan** : “Design one, built it on any platform”
  
  > eg. Deutsche Bank intends to retain the design for about 20 years regardless of the different technological changes
MDA | Models

» PIM (Platform Independent Model) is an abstract model independent from any technology

» PSM (Platform Specific Model) specifies how the functionality specified in a PIM is realized on a given platform.
  > A PIM is transformed into one or more PSMs

» PIMs and PSMs are expressed in UML profiles or metamodels

» The ultimate goal is to generate the system implementation (among other views) by means of model transformations
Model Transformations
» OMG standards which provide a well-established foundation for defining PIMs and PSMs

- UML: Unified Modeling Language
- MOF: Meta Object Facility

<table>
<thead>
<tr>
<th>Meta level</th>
<th>MOF terms</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>M3</td>
<td>Meta-metamodel</td>
<td>MOF models</td>
</tr>
<tr>
<td>M2</td>
<td>Meta-metadata, metamodel</td>
<td>UML Metamodel, UML profiles</td>
</tr>
<tr>
<td>M1</td>
<td>Metada, model</td>
<td>UML Models (eg. Class diagrams)</td>
</tr>
<tr>
<td>M0</td>
<td>Data</td>
<td>Modeled systems</td>
</tr>
</tbody>
</table>

OMG metamodel architecture
Model Transformations

- Source Metamodel
- Transformation Rules
- Target Metamodel
- Source Model
- Transformation Language
- Target Model
- Transformation Engine
Model Transformations

Warning!

Source Metamodel based on Source Model

Transformation Rules based on Transformation Language based on MOF

to

to

to

Transformation Engine exec

Source Model

Target Metamodel based on Target Model

based on
Model Transformations | mappings

» **PIM to PIM**: used when models are enhanced, *refined* or filtered during the development lifecycle without needing any platform dependent information.

» **PIM to PSM**: used when a sufficiently refined PIM is projected to the execution infrastructure.

» **PSM to PSM**: used for component realization and deployment, generally related to platform dependent model refinement.

» **PSM to PIM**: used for mining PIMs from concrete PSMs. Typically called re-engineering and cannot be fully automated, requires renovation tools.
Model Transformations | languages

» Declarative vs imperative matching algorithm

» Unidirectional vs bidirectional
  > Unidirectional transformations are usually imperative
  > Bidirectional transformations are usually declarative, potentially subject to unbounded time execution, problems from a practical standpoint

» Stateless vs persistent
  > Stateless transformations generate each time a new instance
  > Persistent transformations perform the minimum alteration to the target model to propagate the changes leaving the rest intact

» Practical approaches tend to be unidirectional and persistent
Abstract State Machines (1)

» Invented by Y. Gurevich and (promoted by) E. Börger

» ASMs tend to bridge the gap between specification and computation by providing more versatile Turing-complete machines

» ASMs is a variant of first-order logic with equality, where the fundamental concept is that functions are defined over a set $U$ and can be changed point-wise

» Ability to simulate algorithms on their natural levels of abstraction without implementing them

» Extended literature on high-level system design and analysis (see [Börger03])
Abstract State Machines (2)

» Systems of finitely many *transition rules* of form

\[
\text{if } \text{Condition then Updates}
\]

which transform abstract states.

> *Condition*: arbitrary first-order formula without free variables

> *Updates*: finite set of function updates of form \( f(t_1, \ldots, t_n) := t \)

simultaneously executed when *Condition* is true

» A mathematically rigorous form to capture fundamental operational intuitions of computing
Model Transformations

MOF

Transformation Language

Source Metamodel
Source Model
Transformation Rules
Transformation Engine
Target Metamodel
Target Model

Based on
From
Source
Exec
Target
To
Based on
Based on
Based on
ASM and Model Transformations

Source Model \[\xrightarrow{\text{XMI/XSLT}}\] Algebra Source Model

ASMs

Automatic generation

Target Model

Model Transformation Specification

Source Model

Target Model

Algebra Target Model
Model Transformation | an example

» Overall architecture

» Source model
  > Conceptual description of a data-intensive Web application
  > The algebraic encoding

» Target model(s)
  > Model-View-Controller complaint platform-specific model
    > Model
    > View-Controller

» A simple ASM transformation rule
  > A structured content (Web page in the source model) is mapped in the MVC design pattern (controller + view)
Overall Architecture
Webile [IJWET 04] is a UML profile to model data-intensive Web apps.
Source Model | algebraic encoding
Target Model > Model (in the sense of MVC)
Target Model | View-Controller
An ASM rule (1)

asm StructuredContent is
  do forall x in StructuredContent
    extend ServerPage with s1,s2 and ClientPage with c and Build with b and Forward with r and Use with u
    source(b) := s1
    target(b) := c
    source(r) := s2
    target(r) := s1
    source(u) := s2
    target(u) := bd
    controller(x) := s2
    serverView(x) := s1
    clientView(x) := c
  
  generatedFrom({s1,s2,c,b,r}) = {x}

  endextend
enddo
endasm
An ASM rule (2)

\[
\text{asm StructuredContent is} \\
\quad \text{do forall } x \text{ in StructuredContent} \\
\quad \quad \text{extend ServerPage with } s1,s2 \text{ and ClientPage with } c \text{ and Build with } b \text{ and} \\
\quad \quad \quad \text{Forward with } r \text{ and Use with } u \\
\quad \quad \quad \text{source}(b) := s1 \\
\quad \quad \quad \text{target}(b) := c \\
\quad \quad \quad \text{source}(r) := s2 \\
\quad \quad \quad \text{target}(r) := s1 \\
\quad \quad \quad \text{source}(u) := s2 \\
\quad \quad \quad \text{target}(u) := bd \\
\quad \quad \quad \text{controller}(x) := s2 \\
\quad \quad \quad \text{serverView}(x) := s1 \\
\quad \quad \quad \text{clientView}(x) := c \\
\quad \quad \quad \text{generatedFrom}({s1,s2,c,b,r}) = \{x\} \\
\quad \text{endextend} \\
\text{enddo} \\
\text{endasm}
\]
An ASM rule (3)

```
asm StructuredContent is
  do forall x in StructuredContent
    extend ServerPage with s1,s2 and ClientPage with c and Build with b and Forward with r and Use with u
    source(b) := s1
    target(b) := c
    source(r) := s2
    target(r) := s1
    source(u) := s2
    target(u) := bd
    controller(x) := s2
    serverView(x) := s1
    clientView(x) := c
  generatedFrom({s1,s2,c,b,r}) = {x}
endextend
enddo
endasm
```
An ASM rule (4)

asm StructuredContent is
  do forall x in StructuredContent
    extend ServerPage with s1,s2 and
      Forward with r and Use with c
      source(b) := s1
      target(b) := c
      source(r) := s2
      target(r) := s1
      source(u) := s2
      target(u) := bd
      controller(x) := s2
      serverView(x) := s1
      clientView(x) := c
    generatedFrom({s1,s2,c,b,r}) = {x}
  endextend
enddo
endasm

Persistent transformation
Explicit tracking information for change propagation, usually implicit.
Conclusions

» Protecting investment by separating the business model from the supporting technologies

» Model transformations play a key rôle in the OMG's Model Driven Architecture initiative

» Persistent model transformations allow advanced usage scenarios that are currently largely unfeasible [Tratt 04]

» The presented approach to model transformation provides with a flexible, efficient, and practical platform for creating model transformations
Conclusions

» Protecting investment by separating the business model from the supporting technologies

» Model transformations play a key rôle in the OMG's Model Driven Architecture initiative

» Persistent model transformations allow advanced usage scenarios that are currently largely unfeasible [Tratt 04]

» The presented approach to model transformation provides with a flexible, efficient, and practical platform for creating model transformations

» Tool support with Asmatic