Transformational Design with Model-Driven Architecture (MDA)

Prof. Dr. U. Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Gruppe Softwaretechnologie
http://st.inf.tu-dresden.de
Version 08-0.1, June 14, 2008
References

**Obligatory:**
- MDA Guide. OMG (ed.). Reference document for MDA applications

**Optional:**
Many products must be produced in variants for different platforms
- Machines ranging from PDA over PC to host
- Component models from .NET over CORBA to EJB

How to develop a product line?

How to produce common parts of models?
Problem: The Representation Schizophrenia

- **Problem: Design Aging**
  - If an artifact has several representations, such as design, implementation, documentation
  - Always the code is modified, and the other become inconsistent
  - Usually, a design specification ages faster than implementation, because the programmers are tempted to change the implementation quickly, due to deadlines and customer requests
  - They “forget” to update the design

- **Solution:**
  - XP: Single-source principle
    - don’t represent in other ways that code
    - “clean code that works”
  - MDA: do a round-trip to solve the problem
    - One of the biggest problems in software maintenance
Remember: Refinement-based Modelling

- (Old idea. Broadband languages, such as CIP or IPSEN did this in the 70s already)
- Start with some simple model
- Apply refinement steps:
  - Elaborate (more details – change semantics)
  - Add platform-specific details
  - Semantics-preserving operations
    - Restructure (more structure, but keep requirements and delivery, i.e., semantics)
    - Split (decompose, introduce hierarchies, layers, reducibility)
    - Coalesce (rearrange)
    - TransformDomains (change representation, but keep semantics)
- MDA [http://www.OMG.org/mda](http://www.OMG.org/mda) is a refinement-based software development method

- **Split the models into**
  - **Platform-independent model**: The PIM focuses on the logical architecture
  - **Platform-specific model**: The PSM adds platform-specific details and timing constraints

- **Advantages**
  - Separation of concerns: Platform-independent vs platform-dependent issues
  - Portability
  - Automation: derive implementation models from design models (semi-)automatically

- **OMG expects MDA to be their major activity area for the next 5 years**
MDA Describes Product Lines

The platform stack is a *translational framework*

- Domain model for application domain
- Computationally Independent Model (CIM)
  - Requirements specification
- Platform Independent Model (PIM)
- Platform Specific Model (PSM)
- Code

The products of the product line
Example: MDA Performed by Hand

- **Realize active/passive objects**
  - Requirements Specification (UML, formal methods, ...)
  - PIM (standard UML with parallelism)

- **Adaptation to EJB platform**
  - PSM (parallelism resolved)
  - PSM (EJB middleware)

- **Elimination of abstract relations**
  - PSM (relations refined)
  - PSM (.NET middleware)
  - PSM (relations refined)

- **Elimination of all non-Java constructs**
  - PSM (Java Code)
  - PSM (C# Code)
Example: Compilers Are Simple MDA Tools

- Metamodels are language descriptions
- Models are intermediate representations
- Platform specific (abstract syntax tree)
- Platform dependent (binary code)

Programming Language in Concrete Syntax

Abstract Syntax Tree (AST)

Intermediate Language (IL)

Machine Language (ML)
Platforms are environments on which a system runs

- **Abstract machines**
  - Libraries, such as JDK, .NET

- **Implementation languages**
  - Java, Eiffel, C#

- **Component models**
  - CORBA, Enterprise Java Beans (EJB), .NET-COM+, etc.

- **Ontology of a domain (e.g., medicine)**

- **Constraints**
  - Time
  - Memory
  - Energy
- **MDA sees the system development process as a sequence of transformation steps from requirements to code**
  - MDA is an architectural style for transformational frameworks
- **Separation of Platform Information (separation of concerns) reduces dependencies on platform**
  - Middleware (.NET, Corba, DCOM, Beans)
  - Platform specific details (resource constraints, memory handling)
  - Platforms in embedded and realtime systems
  - Domain
- **Reuse of PIM for many platforms**
  - The PIM is a *generic framework* for a product family
  - A *transformational framework*, not an object-oriented framework
- **MDA provides generic frameworks for designs and models**
  - Parameterization with model mappings
The MDA Guide suggests several MDA patterns, i.e., mapping patterns between PIM and PSM:

- **Instantiation**: binding the formal parameters of a template (instantiation of templates, framework instantiation) [see Design Patterns and Frameworks]
- **Isomorphic mapping**: expand a tag in a PIM to n elements of a PSM (1:n mapping)
  - Important to map a class in a PIM to several classes of a component model in a PSM
  - The extension information of a PSM can be expressed as one stereotype in a PIM (marked PIM)
- **Domain transformation mapping**: Change a concept of a PIM into another concept in a PSM
  - For instance, a PIM method to a PSM Command object
- **Aspect mappings**: aspects are woven into the core PIM
  - For instance, with a GRS
1:1 or 1:n mappings (isomorphic mappings, marked PIMs) are important

- They introduce an exclusively-owns relationship from 1 element of the PIM to n elements in the PSM
  - Supported by many UML and MDA tools
- They partition the PIM and the PSM: The border of a partition is demarcated by the PIM tag
- This serve for clear responsibilities, on which level a partition is edited
What Are UML Profiles?

- **Platforms are described by UML profiles**
  - Technically, a profile is a set of new stereotypes and tagged values
  - A profile has a metamodel that extends the UML metamodel
  - Stereotypes are metaclasses in this metamodel that are derived from standard UML metaclasses

- **A profile can be a domain model**
  - or ontology, if domain is large enough

- **A profile can be a domain specific language (DSL)**
  - With own vocabulary
  - Every entry in metamodel is a term

- **Examples**
  - EDOC Enterprise Distributed Objects Computing
  - Middleware: Corba, .NET, EJB
  - Embedded and realtime systems: time, performance, schedulability
Marking

PIM

marked PIM

Transformation

Marks

Mapping

Platform

PSM

[MDA Guide, OMG]
Different class implementations in a PSM, refining to different languages, using different patterns

**Java**

```java
public void withdraw(
    int amount) {
    sum -= amount;
}
```

```java
// Java implementation as a decorator
class Loan extends Account {
    // decorator backlink
    Account upper;

    private int sum;
    public void withdraw(
        int amount) {
        sum -= amount;
    }
}
```

**C#**

```csharp
public void withdraw(
    int amount) {
    sum -= amount;
}
```

```csharp
// C# implementation: a partial class
class Loan partial Account {
    private int sum;
    public void withdraw(
        int amount) {
        sum -= amount;
    }
}
```
Meta Model Transformation

PIM → Platform Independent Metamodel

Transformation Specification

PSM → Platform Specific Metamodel

[MDA Guide, OMG]
Model Transformation

```
PIM  ____________  Platform Independent Types
    |  subtypes of
    v

Transformation Specification

Transformation

PSM  ____________  Platform Specific Types
    |  subtypes of

[MDA Guide, OMG]
```
Pattern Transformation

PIM

marked PIM

Transformation

Pattern Names

Patterns

Platform

PSM

[MDA Guide, OMG]
Model Merging

- PIM
- Another Model
- Model Merge
- PSM

[MDA Guide, OMG]
Additional Information

[MDA Guide, OMG]
When Can We Semi-Automatically Enrich A PIM to a PSM?

- Describe *platform specific extension (PSE) as aspects or views*
- The PIM is the *core*, the PSM the *weaved system*
- The model mapping becomes an *aspect weaver*
MDA With Several Layers for Resource-Constrained Systems

- **HIDOORS EU Projekt**
  (High Integrity Distributed Object-Oriented Real-Time Systems),
  http://www.hidoors.org

- **MDA for RT-UML**
  - Realtime sequence diagrams (MSC)
  - UML realtime statecharts

- **Transformation into timed automata of Uppaal model checker**
RT Sequence Diagram (UML)

RT Extension Aspect

<<subject>> Heart Rate Server
<<observer>> HR Trend Recorder
<<observer>> HR Sensor

Join Points

A
GetRate()

B

C

D
GetRate(
{D-C<=1ms}
{B-A <= 2ms}
)

Advice:

Subscribe()
RT-SD und RT-Statecharts are Platform Specific Aspects

PIM: UML class diagram

RT Sequence diagram

PSM-1

RT-Statecharts

PSM-2
Problem: Full MDA Needs Roundtrip

- Otherwise, the models age (design aging)
- This is still an unsolved problem

Model Mappings

Requirements Specification

Platform Independent Model (PIM)

Platform Specific Model (PSM)

Code
Problem 2: MDA Needs More Levels (Multi-Stage MDA)

- Requirements Specification
- Platform Independent Model (PIM)
- Platform Specific Model (PSM)
- Code

"platform stack"
Model-based software development (MDSD, MDD) tags UML diagrams with *domain profiles*
- From the profile stereotypes and tags, domain-specific code is generated
- set/get, standard functions, standard attributes
- compliance functions for component models

**In contrast, MDA profile tags are platform-specific**

class Loan extends IAccount {
    private Person owner;
    void setOwner(Person p) {..}
    Person getOwner() {..}
    private int sum;
    /*** end generated code ***/
    public void withdraw(
        int amount) {
        sum -= amount;
    }
    /*** begin generated code ***/
    }

<<Account>>
Loan
withdraw()
MDA(R) is a trademark of OMG