30 Transformational Design with Essential Aspect Decomposition: Model-Driven Architecture (MDA)

1. Model-Driven Architecture
2. Model Mappings
3. Model Merging and Weaving
4. MDSD with domain-specific tagging
Obligatory:
- www.omg.org/mda Model driven architecture.
- 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
30.1 MODEL-DRIVEN ARCHITECTURE
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 is a refinement-based software development method for product families (product lines)

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
- **Platform-specific implementation**: contains the code
- **Platform description model**: describes the platform concepts

Advantages

- Separation of concerns: Platform-independent vs platform-dependent issues
- Portability
- Automation: derive implementation models from design models (semi-) automatically
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)
- Platform-Specific Implementation (PSI, Code)

The products of the product line
Model mappings connect models horizontally (on the same level) or vertically (crossing levels).
- From a model mapping, a simple transformation can be inferred

Model weavings weave two input models to an output model
- Usually, some parts are still hand-written code

- Domain model for application domain
- Computationally Independent Model (CIM) Requirements specification
- Platform Independent Model (PIM)
- Platform Description Model (PDM)
- Platform Specific Model (PSM)
- Platform-Specific Implementation (PSI, Code)
- Handwritten code
Example: MDA Performed by Hand

- **Requirements Specification** (UML, formal methods, ...)
- **Realize active/passive objects**
  - PIM (standard UML with parallelism)
- **Adaptation to EJB platform**
  - PSM (parallelism resolved)
- **Elimination of abstract relations**
  - PSM (EJB middleware)
  - PSM (relations refined)
  - PSM (.NET middleware)
- **Elimination of all non-Java constructs**
  - PSM (Java Code)
  - PSM (relations refined)
  - PSM (C# Code)

TU Dresden, Prof. U. Aßmann
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 Description Model (PDM)

Machine Language (ML)
What are Model Mappings?

- **Model**
  - “A model is a representation of a part of a function of a system, its structure, or behavior”
- **Model mappings are transformations from an upper to a lower model**
  - The mappings are automatic or semi-automatic: step-wise refinement of the model by transformation

![Diagram of model mappings](Diagram.png)
Platforms are *variability levels*, variants that produce a variant of the specification.

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
Benefit of MDA

- 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
30.2 MODEL MAPPINGS
Different Kinds of 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 (1:1 mapping)
  - Important to map a element of a PIM to several elements of a PSM
  - The extension information of a PSM can be expressed as one stereotype in a PIM (marked PIM)
- **Homomorphic mapping**: expand a tag in a PIM to \( n \) elements of a PSM (1:n mapping)
  - Important to map a element of a PIM to several elements of a PSM
  - The extension information of a PSM can be expressed as one stereotype in a PIM (marked PIM)
- **Concept 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?

- A **UML** **profile** is a metamodel describing a platforms or a domain
  - Technically, a profile is a set of new stereotypes and tagged values
  - Stereotypes correspond to metaclasses
  - A profile has a metamodel that extends the UML metamodel
  - Stereotypes are metaclasses in this metamodel that are derived from standard UML metaclasses

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

- A **profile** can describe a domain model
  - or ontology, if domain is large enough
  - A **profile** can be the core of a domain specific language (DSL)
  - With own vocabulary, every entry in metamodel is a term

- Examples:
  - Banking, insurances, cars, airplanes, ...
Marking

PIM

marked PIM

Transformation

Marks

Mapping

Platform

PSM

[MDA Guide, OMG]
Example of a Marked PIM

- Different class implementations in a PSM, refining to different languages, using different patterns

```java
<<Java>>
Loan
- int sum
+ withdraw()

Loan
- int sum
+ withdraw()

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

```csharp
<<C#>>

Loan
- int sum
+ withdraw()

Loan partial Account
- private int sum;
+ public void withdraw(int amount) {
    sum -= amount;
}
```

// Java implementation as a decorator
class Loan extends Account {
    // decorator backlink
    Account upper;
    
    private int sum;
    public void withdraw(int amount) {
        sum -= amount;
    }
}

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

[MDA Guide, OMG]
Model Transformation

TU Dresden, Prof. U. Aßmann

[MDA Guide, OMG]
Meta Model Transformation

[MDA Guide, OMG]
30.3 Model Merging and Weaving

PIM -> Model Merge -> PSM

Another Model

[MDA Guide, OMG]
Additional Information

[MDA Guide, OMG]
Adding Platform-Specific Extensions to Platform-Independent Models

- Essence
- Platform independent model (PIM)
- Platform-1 specific model (PSM)
- Platform-1 specific extension (PSE)
- Platform-2 specific model (PSM)
- Platform-2 specific extension (PSE)
- Platform-(1+2) specific model (PSM)

Model weaving

Aspect 1

Aspect 2
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*.

Diagram:

- Top level:
PIM

- Middle level:
  - PSE for Aspect 1
  - PSE for Aspect 2
  - PSE for Aspect 3

- Bottom level:
  - PSM
- 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

Subscribe()

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

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)
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-->

```java
public void withdraw(int amount) {
    sum -= amount;
}
```
MDA(R) is a trademark of OMG