Model-Driven Architecture (MDA)

Literature:
A. Kleppe, J. Warmer, W. Bast:
Addison-Wesley, 2003. Available for students as electronic copy in Kvartersbibliotek B.
S. Mellor, K. Scott, A. Uhl, D. Weise:
OMG: www.omg.org/mda

Model
- Set of elements that describes some physical, abstract, or hypothetical system
  - Abstraction from one or several properties
    - E.g., real size, material, level of detail
  - Means of communication
  - Cheaper to build than the real system
  - More suitable for analysis (e.g., by static analysis or simulation) than the real system
  - Decision help
- Examples:
  - Modelica models for physical/technical systems
  - UML model = abstraction of a program (software system)
    - Level of detail can vary from coarse-grained blocks to executable code

Platform
- Specification of an execution environment for a set of models
  - E.g.: CORBA, EJB;
    - Java JVM, C++; Linux, Solaris, Windows, RTOS;
    - SPARC, IA-64, PowerPC;
    - VHDL; …
  - Needs to have at least one implementation
    - Which can build upon one or more other platforms (composed realization)
    - Or stand alone (primitive realization)

Mapping between models
- MDA: Iterative and incremental development by successive model refinement, up to code generation
- Mapping:
  - Described at the metamodel level
  - Applicable to all source models that conform to the metamodel
- Can be automated by providing an executable specification
- In full generality not completely automatizable
  - UML only semi-formal, not really executable
  - Needs manual editing for complementation

Marking models
- Marks = light-weight, non-intrusive, persistent extensions to models that capture information required for model transformations without polluting these models
  - "sticky notes" attached to model elements
  - Specific to a mapping
- in UML?
  - Hungarian notation for special class names etc.
  - Special elements defined by a marking model (specified as UML extension)
Background: Customizing UML

- E.g., to construct MDA marking models

Two UML Extension Mechanisms:

- **Stereotypes**
  - Type qualifiers to customize existing language elements (e.g., classes, associations)
  - Example: Definition and use of stereotype <<persistent>>:

```
Class <<stereotype>> persistent <<persistent>> Account
```

- **MOF**

  Can also introduce new graphical symbols in both cases

MOF (Meta-Object Facility)

- The language in which UML is specified
  - A subset of UML itself: reified
    - Types (classes, primitives, types, enumerations)
    - Generalizations (inheritance)
    - Attributes
    - Associations
    - Operations
  - Example: Definition of a UML comment in MOF:

```
MOF specifies only structural and behavioral aspects
- Not how to store, graphically represent, or edit UML models
  - left to tool providers
- Except for an XML-based metadata interchange format: XMI
```

Marking model example

- Definition of a mark element with attributes

```
Class [required] <<stereotype>> JavaMarks
  identifier: String
  synchronized: Boolean
```

UML Profiles

- Collection of stereotypes and metamodel extensions for a special domain or platform
  - Creates an UML dialect
  - Needs standardization

- Examples:
  - UML Real-time profile
  - UML profile for CORBA

PIM, PSM

- **PIM** (Platform-independent model)
  - **PSM** (Platform-specific model)
    - created from a marked PIM by mapping (+ manual complementation afterwards)
  - Marking a PIM for different mappings leads to different PSM’s
  - Can be iterated
  - Code generation from last PSM

PIM, PSM, Model transformations
**Model-driven software development**

- **Requirements**
  - Mostly text
- **Analysis**
- **Low-level design**
- **Coding**
- **Testing**
- **Deployment**

**PIM**:
- Business-oriented, abstracts from platform issues, survives technology changes

**PSM**:
- (Semi-)automatic code generation

**Code**:
- (Semi-)automatic transformation

- This could involve multiple model refinement steps...

**Formal model, e.g. in UML + profile(s)**

**Example: Velocity**
http://velocity.apache.org

**Velocity Template Language** (for static metaprogramming)

- Generate Java source code from Java-PSM in UML

Example template in Velocity, generating a simple Java class:

```java
#classModifiers ( $class )
class $class.name {
#
#foreach ( $field in $class.fields )
#fieldModifiers ( $field )  $field.type.name $field.name ;
#
#end
#
#foreach ( $constructor in $class.constructors )
#constructorModifiers ( $constructor )  $class.name ( … )  {  }
#
#end
#
#foreach ( $method in $class.methods )
…
#
#end}
```

**MDA vs. MDA-light**

**OMG-MDA**:
- Clean separation of business logic and platform issues ⊗
- More reuse potential ⊘
- Better maintainable, debuggable ⊘
- Still under development ⊗

<table>
<thead>
<tr>
<th>MDA-light</th>
</tr>
</thead>
</table>
| Long way to go in one shot ⊘
| PIM polluted with marks for low-level technical stuff ⊘
| Works today in practice ⊗
| E.g. xtUML framework at Saab |

| PIM |
| PIM |
| PSM |
| Model compiler AND code generator in one |

**Code**

**Summary: MDA**

- Increased reuse
- Increased programmer productivity
- Relies on good tools:
  - Free: Eclipse EMF, GME (ISIS, Vanderbilt U.) for Visual Studio.NET, GMT for Eclipse
  - IBM MTF, OpenMDX (www.openmdx.org), UMT, ...
  - And many commercial ones, e.g. Telelogic TAU
- Still in its infancy
- Could become the mainstream software engineering technology by 2020
- Consistency problem: How to map manual edits in PSM or generated code back to a source model?
- Automatic Roundtrip Engineering (ARE)