TDDD05
Component-Based Software

Model-Driven Architecture (MDA)

Literature:

A. Kleppe, J. Warmer, W. Bast:
MDA Explained: The Model Driven Architecture (TM): Practice and Promise.
Addison-Wesley, 2003. Available for students as electronic copy in Kvartersbibliotek B.

S. Mellor, K. Scott, A. Uhl, D. Weise:
MDA Distilled – Principles of Model-Driven Architecture. Addison-Wesley, 2004.

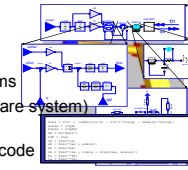
OMG: www.omg.org/mda

Christoph Kessler, IDA,
Linköpings universitet, 2010.

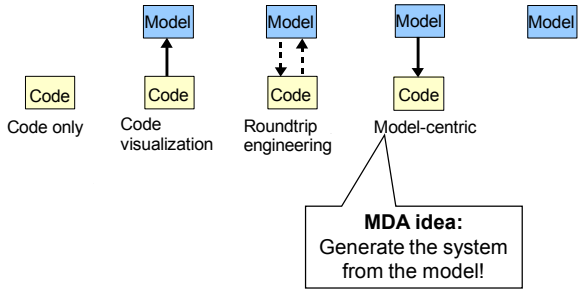
Model

- 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

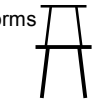


Relations between model and code



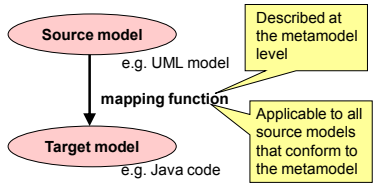
Platform

- 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

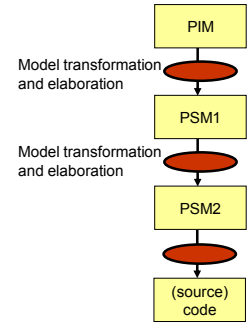
- **MDA**: Iterative and incremental development by *successive model refinement*, up to code generation
- **Mapping**:



  - 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

PIM, PSM

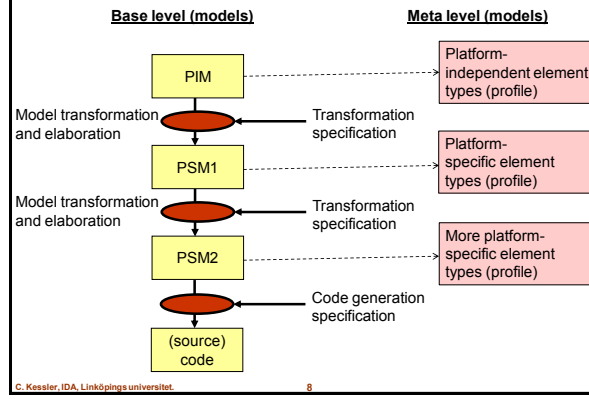
- **PIM** (Platform-independent model)
  - business-oriented,
  - Abstracts from platform issues
  - survives change of platform
- **PSM** (Platform-specific model)
  - contains platform specific modeling elements, types etc.
- Can be iterated
- Code generation from last PSM



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, Model transformations

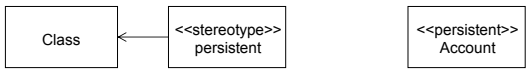


Background: Customizing UML

- 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>>:

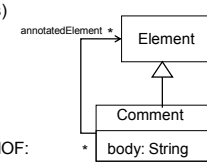


- **MOF**

Can also introduce new graphical symbols in both cases

Background: MOF (Meta-Object Facility)

- MOF = the language in which UML is specified
- A subset of UML itself: (→ reified)
  - Types
    (classes, primitives, types, enumerations)
  - Generalizations (inheritance)
  - Attributes
  - Associations
  - Operations



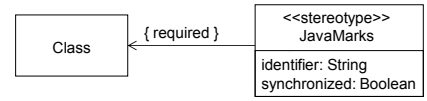
- Example: Definition of a UML comment in MOF:
- MOF specifies only structural and behavioral aspects
  - Not how to store, graphically represent, or edit UML models – left to tool providers
  - Except for an XML-based metadata interchange format: XMI
- Use MOF for fundamental extensions of the UML language

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)

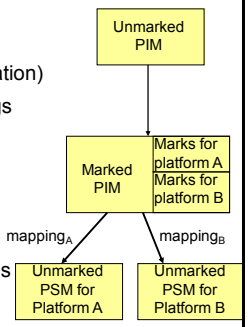
Marking model example

- Definition of a mark element with attributes
  - Example: as a stereotype of the Class metaclass



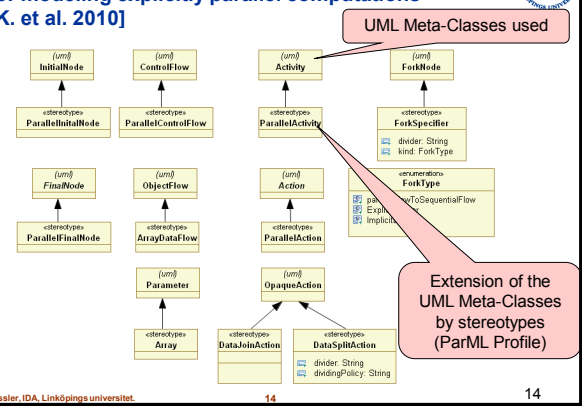
Using marking models to guide model transformations and code generation

- PSM created from a marked PIM by mapping + manual complementation (elaboration)
- Marking a PIM for different mappings leads to different PSM's
- The marking is added manually (eg. using a special UML editor)
- The marking serves as possible anchor points (cf. declared hooks) for automated model transformations
- This process can be iterated
- Code generation from last PSM



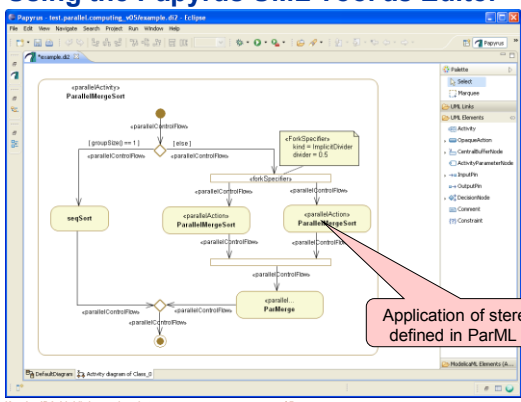
Example: ParML – a UML profile

for modeling explicitly parallel computations [K. et al. 2010]



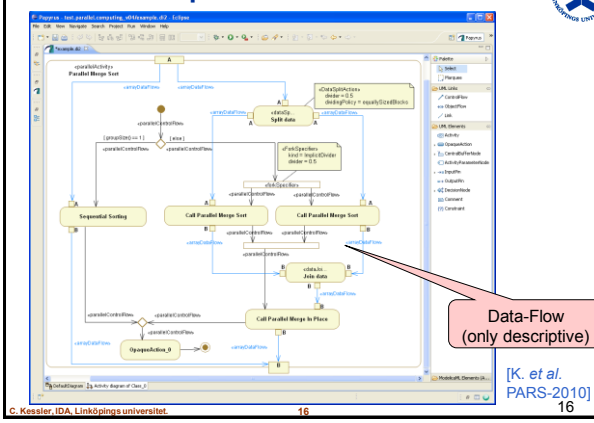
14

Using the Papyrus UML Tool as Editor



15

ParML Example: Control- and Data-Flow



[K. *et al.* PARS-2010]

16

Source code generation with templates (boilerplates, code skeletons)

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

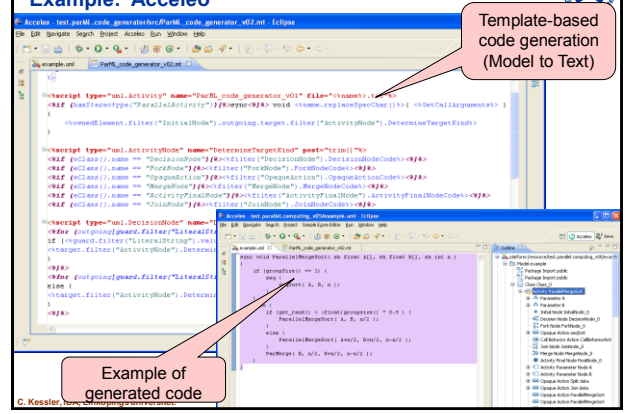
- Velocity Template Language (for static metaprogramming)
- Generate Java source code from Java-PSM in UML

```
#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
}
```

Example template in Velocity, generating a simple Java class

Source code generation with templates

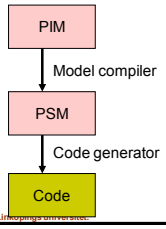
Example: Acceleo



MDA vs. MDA-light

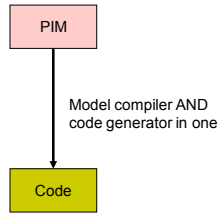
OMG-MDA®:

- Clean separation of business logic and platform issues ☺
- More reuse potential ☺ better maintainable, debuggable
- Still under development ☹



MDA-light

- 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



Summary: MDA

- **Increased reuse**
  - PIM survives change of platform
- **Increased programmer productivity**
  - Part of the code is generated automagically, hence less code to be written by hand
- **Semi-automatic**
- **Relies on good tools:**
  Model editors, model repositories, model transformers, code generators
  - free: Eclipse EMF, GME (ISIS, Vanderbilt U.) for Visual Studio.NET, GMT for Eclipse, IBM MTF, OpenMDX (www.openmdx.org), UMT, Papyrus, Acceleo...
  - 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)

Further References

- C. Kessler, W. Schamai, P. Fritzson: Platform-Independent Modeling of Explicitly Parallel Programs. Proc. PARS-Workshop at ARCS-2010, Hannover, Germany, VDE-Verlag, 2010
  - A case study of extending UML for modeling explicitly parallel computations, using open-source MDA tools.
  - www.ida.liu.se/~chrke/publ.html