Model Driven Architecture
Model Driven Development

Johann Oberleitner
joe@infosys.tuwien.ac.at
23. 5. 2006

Agenda

- Motivation
- Goals of MDA
- Technologies/Paradigms for MDA
- MDA Tools
- MDA – State of the Art
- Next Steps

Demands on Software Industry

- User demands on Software Industry
  - Productivity
    - Fast development
    - More features
    - Even more features
  - Quality
    - Bugfree software
  - Longevity
    - I want this software for that platform NOW
    - It want to use all features of that platform

Pressures to SW Industry Today / Productivity

- Simple changes
  - Effects In multiple tiers
    - Requires test or verification of whole system
  - Example: addition of a new data attribute
    - Database table
    - Server tier
    - Client tier
    - User interface
  - Even worse when used by other systems

Pressures to SW Industry Today / Quality

- Tendency
  - Code/Coding is everything
  - Architecture/Design less important
  - Documentation is done when there is time for it
    - Gap between documentation and code
  - Testing when there is time for it
  - Testing is done by the end user
  - Bugs in standard products are only tip of the iceberg
    - Interaction with many other products
  - Independent of Open/Closed Source
    - Prominent open source projects have many hands
- Not possible in other Engineering Fields
  - High-rise buildings
    - Structurally sound to resist wind, rain, earthquakes
    - Miles of heating and cooling system ducts, telephone, and network cable, and power conduits
  - Imagine what happens if high-rise buildings are constructed with quality of software

Pressures to SW Industry Today / Longevity

- Every 5 years new platforms/technologies/paradigms introduced
- J2EE, .NET, Web Services
  - None of these platforms (practically) available 5 years ago
- Java today different from Java 1.1
  - Includes J2EE, J2SE, J2ME
  - AWT, Swing, SWT
  - Java 5 includes Generics & other features (some from .NET)
- Web
  - JSP,ASP.NET,PHP
- XML
  - XML DTD to XML Schema
  - WSDL, SOAP, UDDI
- CORBA
  - CCM and EJB?
- Microsoft
  - DCOM changes to COM+ changes to .NET
  - 2005: .NET 2.0

Attempt to Encounter Pressures

Solution that worked before:

- ***Raising Level of Abstraction***

- Machine-code to Assembly language
- Assembly-Language to higher languages
  - Fortran, COBOL
  - Pascal, C
  - C++,Java,C#,Eiffel
- Operating Systems
  - Provide APIs for accessing lower-level constructs

Model Driven Architecture Main Goal

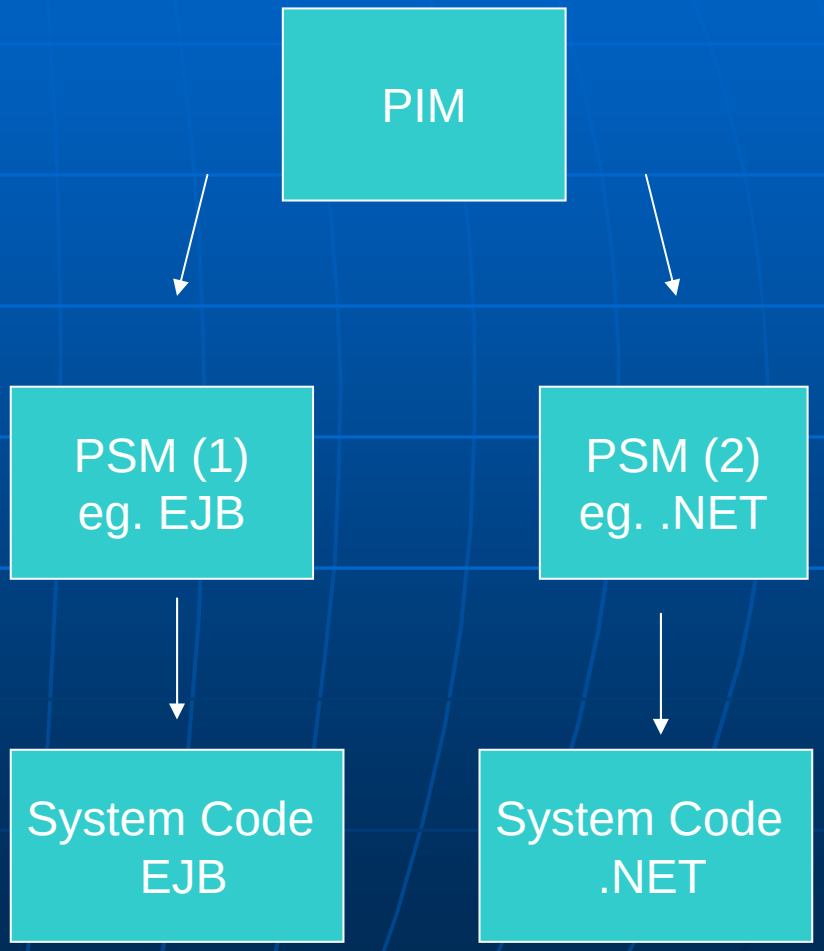
- Applications are constructed from models
  - Less implementation effort
- More realistically
  - Large parts of applications are constructed from models
  - Still less implementation effort

Model Driven Architecture

- Raising Abstraction Level to Models
- Whole system is represented in a model
  - Successor of Case Tools of the Nineties
- Code generators generate executable systems
- MDA
  - OMG (CORBA, UML)
- Model Driven Development
  - More generic, not OMG specific

Primary Idea of MDA / 1

- Platform Independent Model (PIM)
- PIM has role of source code
- Platform Specific Models (PSM)
- Platform code generated from PSM



Primary Idea of MDA / 2

- Platform Specific Models
  - PSMs are generated from PIMs
    - Model transformation
    - Upcoming UML Standard (QVT)
    - Mark Model in PIM
  - Currently only partial PSMs are generated
    - Manual completion of these models
    - Goal/Vision is to generate complete PSMs and complete systems from PIM
  - PSMs are source for code generators
    - These code generators can be complete

Primary Idea of MDA / 3

- Verification of system properties at PIM
  - Formal verification
  - Testing
- Verification of system properties at PSM
  - Formal verification
  - Testing
  - Required Platform specific properties
- In theory no component testing at code level
  - Only System Test
- Documentation always up-to-date

MDA Goals

- Decrease in production costs
  - Many lines of code are represented by model elements
- Longevity
  - Porting effort minimized by switching to different platform
  - Vision is that different MDA tools might interoperate
- Quality
  - Due raise of abstraction level
  - Verification/Testing at PIM level

Technologies

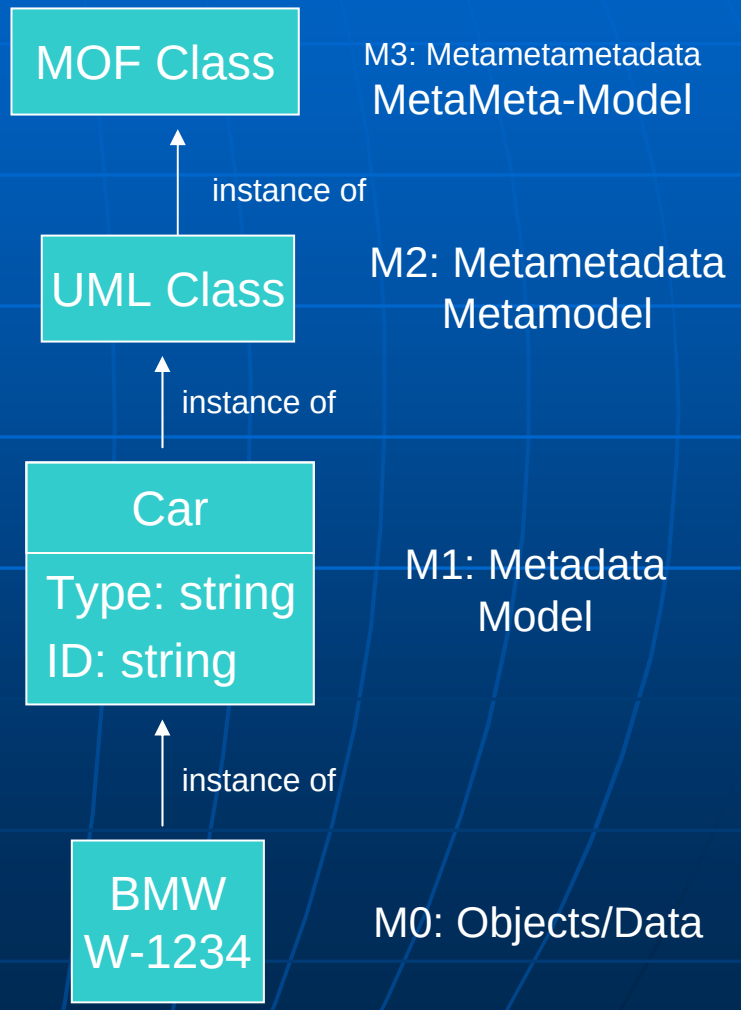
- UML, CWM, MOF
  - Modeling Baseline
- XMI
  - Representation for MOF/UML models
- JMI, EMF
  - APIs for XMI
- OCL
  - Enhances models with formal descriptions
- QVT

MDA Models

- Usually represented in UML (OMG)
  - All MOF based models possible
- Detailed Models
  - Complete naming of elements
  - All details left out in the model are left out in the system
  - Constraints of model elements

Meta-Object Facility (MOF) / 1

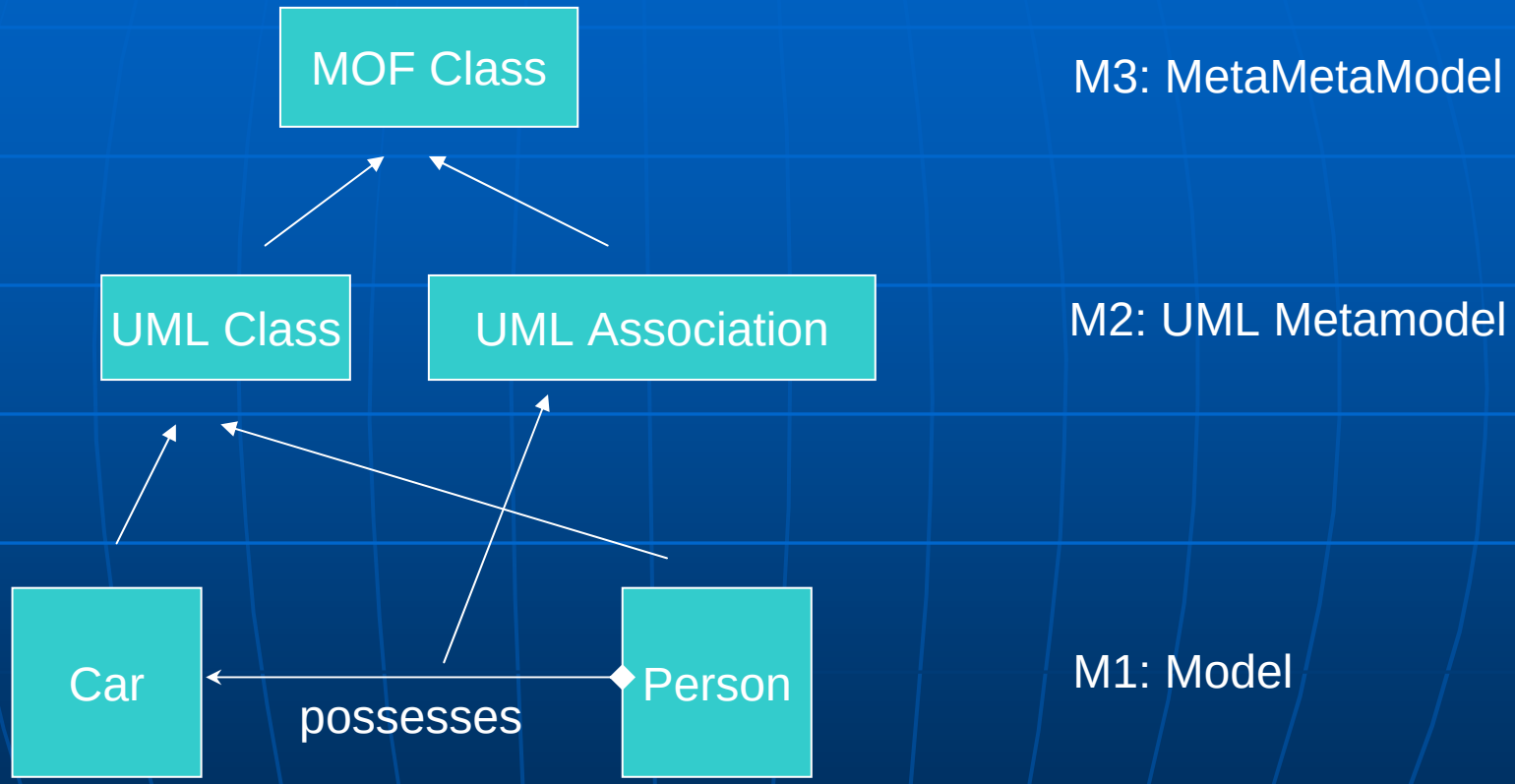
- 4 layer stack
  - Each layer describes model elements
- Base for different modeling standards
  - UML, CWM
  - Principally open
- M3 contains a principal set of modeling constructs
  - MOF Class
  - MOF Association
  - MOF Data Types
  - MOF Exceptions
  - MOF Constants
  - MOF Constraints



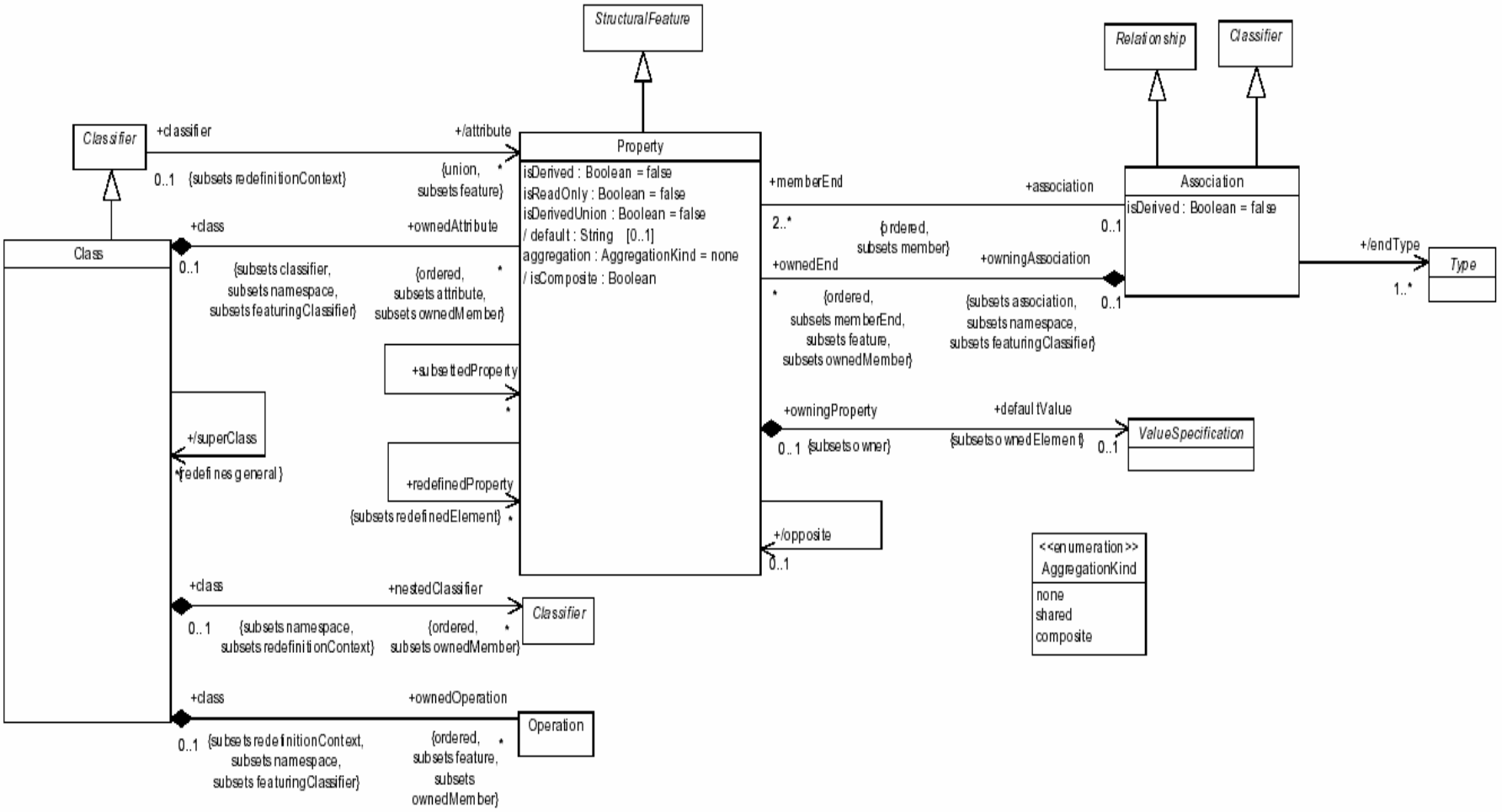
Meta-Object Facility (MOF) / 2

- Metamodels at M2
  - UML, CWM
- UML metamodel
  - Describes syntax of UML models
  - Which modeling elements UML has
    - Instance of MOF class for each UML modeling element
  - Possible Relationships between UML model elements
    - Instance of MOF association for each relationship
  - Constraints/Rules UML models have to adhere to

Meta-Object Facility (MOF) / 3



UML 2 Meta-Model (part)



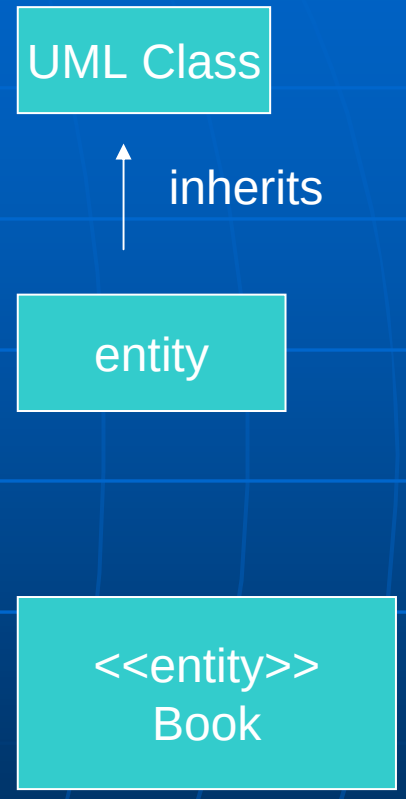
Taken from OMG UML2 Superstructure, Figure 30

UML Extensions

- UML cannot be complete
  - Not feasible to specify every detail
  - Too generic
- 2 ways to extend UML/MOF
  - heavyweight: completely new meta-model
    - Not automatically supported by modeling tools
  - Lightweight:
    - Stereotypes
      - Inherit from constructs at M2
      - Specific semantics
    - Tagged Values
      - Special Values attached to modeling elements
    - UML/MOF profiles
      - Sets of Stereotypes and Tagged Values
      - Modeling Tools can support Profiles

UML Profiles in MDA / 1

- Stereotypes
  - Represented with << ... >>
- Example
  - Book is a class
    - Stereotype inherits from UML Class
  - Book is a persistent entity
    - Has a representation in a database
  - Entity is a new construct
    - Inherits from UML class
  - Book objects can be persisted
    - Eg. in a Database Book table



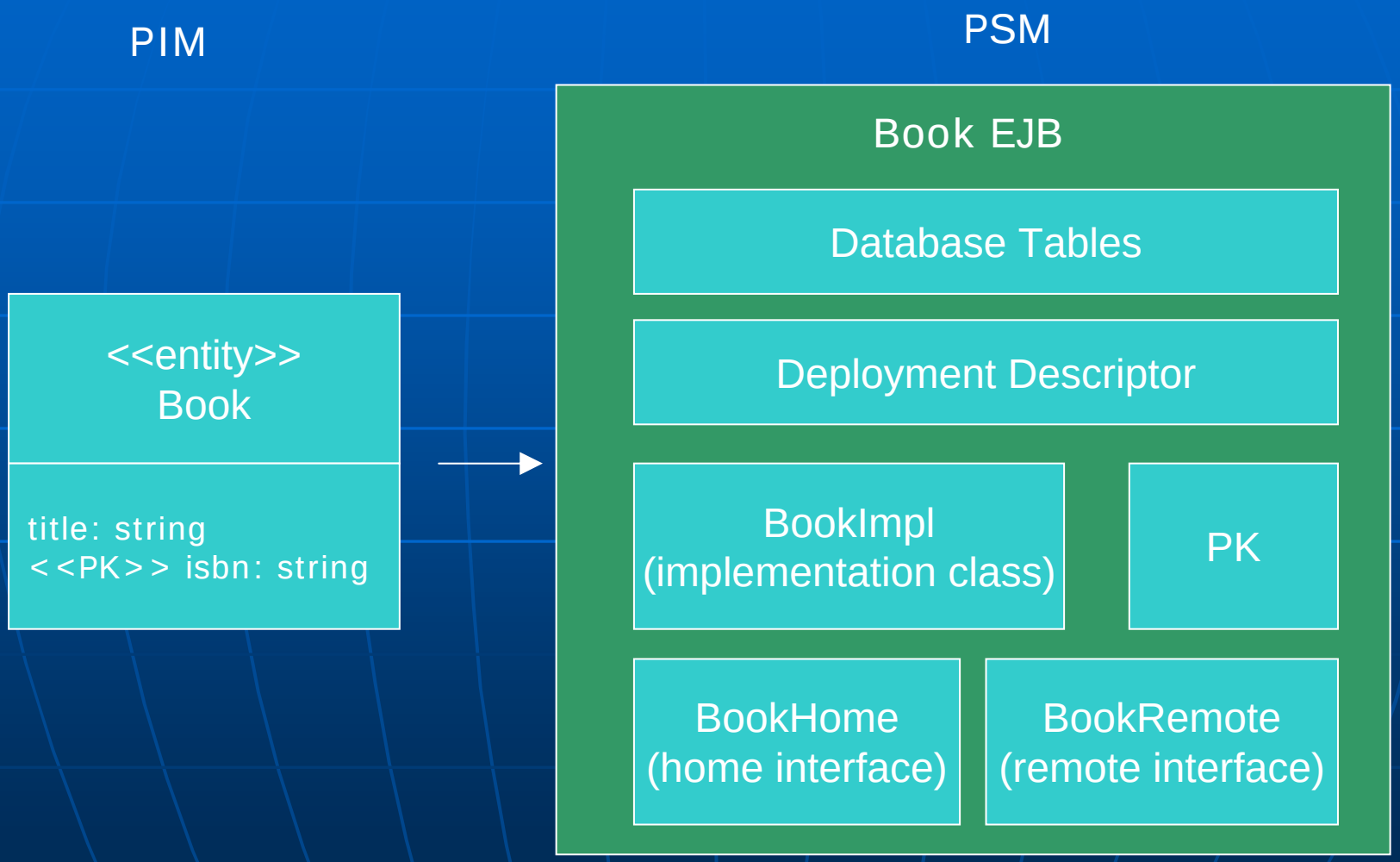
UML Profiles in MDA / 2

- UML Profile for EDOC (Enterprise Distributed Object Computing)
  - Entities, Events, Flows
- Specialized Profiles
  - CORBA, CCM, EJB, ...
- Problem:
  - Only proof-of-concept
  - Incomplete
  - Lack of formality
- MDA Tool Vendors have invented their own profiles
  - Lack of portability

UML Profiles in MDA / 3

- Stereotypes/Tagged Values
  - Interpreted by code generators
- Example
  - Model generator for EJB
    - Stereotype entity
      - persistent objects ~ Entity Beans
    - Stereotype PK
      - Marks attribute as Primary Key

UML Profiles in MDA / 4



UML Action Semantics / 1

- Used for description of actions
  - Very high abstraction level
  - Actions may be reordered
- Programming languages may conform to this semantics
  - Represented in text form
  - Or in graphical form
- Actions
  - take input values (on input pins)
  - can produce output values (at output pins)
- Description
  - flow of control
  - flow of data

UML Action Semantics / 2

- Composite Actions
  - Grouping
  - Conditional Action
  - Loops
- Read/Write actions
  - Set/Read values
  - Object actions
    - Create/destroy objects
  - Selection among objects (similar to SQL SELECT)
- Computation actions
- Collection actions
  - Iterators, filter, ...

UML Action Semantics / 3

- Messaging actions
  - Object invocation, ...
  - Argument values, return Values, ...
- Jump actions
  - Break/continue
  - Exceptions

UML Action Semantics / 4

- Example

```
foreach b in Book[.copyright > 1995
    and copyright < 2002]
{
    b.bookPrice *= 1.2;
 }
```

- Languages
  - Action Specification Language (ASL)
  - BridgePoint Action Language (AL)
  - Kabira Action Semantics (Kabira AS)
  - TALL
  - SMALL
- Different Syntaxes(!)

MOF, UML, CWM

- UML
  - Primary MDA modeling language
  - Many tools available
  - Extensible by profiles
- MOF
  - Baseline of UML
- CWM (Common Warehouse Metamodel)
  - Modeling constructs for Data Warehouses
  - Database Models
  - Query Facilities
  - Supported by major Portal Vendors
    - IBM, Oracle, Unisys

XML Metadata Interchange (XMI) / 1

- XML Metadata Interchange
  - Persistence of MOF-based Models in XML
  - UML/CWM based on MOF -> XMI for UML
- XMI Format for a Model
  - Automatically deduced from its Metamodel
  - No need to write specific parsers
  - One parser supports all XMI formats
- UML2
  - Different XMI format
  - UML diagrams as Scalable Vector Graphics (SVG) within XMI files

XML Metadata Interchange (XMI) / 2

```
<XMI xmi.version = '1.2' xmlns:UML = 'org.omg.xmi.namespace.UML'>
  <XMI.content>
    <UML:Model xmi.id = 'ID-1' ... >
      <UML:Class xmi.id = 'ID-2'
           name = 'Book'
           visibility = 'public'
           isAbstract = 'false'>
        <UML:ModelElement.stereotype>
           <UML:Stereotype xmi.idref = 'ID-777'/>
        </UML:ModelElement.stereotype>
        <UML:Classifier.feature>
           <!-- ... Next slide ...
        </UML:Classifier.feature>
      </UML:Class>
      <UML:Stereotype xmi.id = `ID-777` name = `entity`>
         <UML:Stereotype.baseClass>Class</UML:Stereotype.baseClass/>
      </UML:Stereotype>
    </UML:Model
  </XMI.content>
```

XML Metadata Interchange (XMI) / 3

```
<UML:Classifier.feature>
   <UML:Attribute xmi.id = `ID-10`
       name = `title` visibility = `public`
       ownerScope = `instance`>
     <UML:StructuralFeature.type>
           <UML:Class xmi.idref = `ID-800`/> // ID-800 -> String
     </UML:StructuralFeature.type>
   </UML:Attribute>
   <UML:Attribute xmi.id = `ID-11`
       name = 'isbn' visibility = 'public`
       ownerScope = `instance`>
     <UML:ModelElement.stereotype>
       <UML:Stereotype xmi.idref = `ID-778`/> // ID-778 -> PK
     </UML:ModelElement.stereotype>
     <UML:StructuralFeature.type>
           <UML:Class xmi.idref = `ID-800`/>
     </UML:StructuralFeature.type>
   </UML:Attribute>
</UML:Classifier.feature>
```

XML Metadata Interchange (XMI) / 4

- Disadvantages of XMI
  - Synthesized format
    - Not optimized for UML
  - Files tend to get (very) large
    - 20 classes ~ 250 KB
    - 20 classes + diagrams (SVG) ~ 3 MB
  - Different XMI dialects (?)
    - Incompatibilities between tools
    - Tools have multiple XMI importers/exporters

XMI APIs / 1

- Java Metadata Interface (JMI)
  - MOF Metadata
    - Creation, Storage, Access, Discovery, Exchange
  - Reference implementation (Unisys)
  - Netbeans implementation (Sun)

XMI APIs / 2

- Eclipse Modeling Framework (EMF)
  - Class Diagrams
  - Synchronization of
    - Java source code
    - XMI
    - XML Schema
  - Different from JMI
  - Eclipse Modeling Plugins based on EMF

OCL / 1

- Object Constraint Language
- Used for exact modeling of UML/MOF models
  - Small language
    - Logical & arithmetic operators
    - Property Call & navigation expressions
  - Simple built-in types
  - Collection types
  - Elements of the model can be used

OCL / 2

- Describes Logical Constraints
- Attached to modeling elements
- Documentation Purposes
- Used for formal verification
- Used for generation of testcases
  - Manual and automatic
- Can be checked at runtime
  - May throw exceptions
  - See Design by Contract

OCL / 3

- OCL for pre-/postcondition
- Directly from specification
- Example
  - Class Account models bank account
    - Field balance stores balance
  - Methods deposit/withdraw

```
context Account::deposit(int amount)
pre: amount >= 0
post: balance = balance@pre + amount

context Account::withdraw(int amount)
pre: amount >= 0 and balance >= amount
post: balance = balance@pre - amount
```

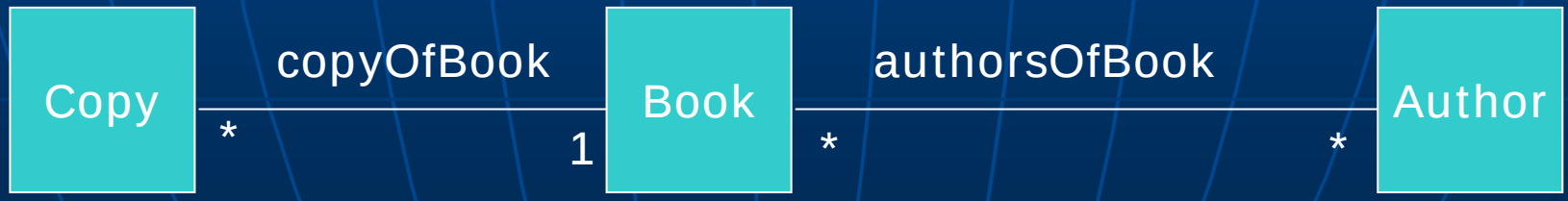
OCL / 4

- OCL used for invariants
  - Holds during lifetime of an object
- Constraints between model elements
  - Defined at the level of the meta-model
  - Must hold before and after transformation

OCL / 5

- OCL for Initialization code
  - Describes outcome of constructors
- OCL for Navigation expressions

```
context Copy
inv: self.copyOfBook.authorsOfBook->size() > 0
```



QVT

- Query, View, Transformation
  - Queries of Models
  - Views (similar to databases) of Models
  - Transformations of Models
- RFC of OMG for MDA transformations
  - ~ 15 submissions
  - Declarative transformation languages
    - Relationships of source & target described with rules
  - Imperative transformation languages
    - Describes explicity each manipulation step
  - Hybrid

MDA Tools

- IDEs & Modeling Tools
  - IBM Rational XDE (Java & .NET)
  - Borland Together
  - Microsoft Visual Studio Team System Architect Edition
  - Poseidon
- MDA based Code Generators
  - AndroMDA
  - VMTE
- Architectural IDEs
  - Interactive Objects ArcStyler

IDEs & Modeling Tools / 1

- Focus on Roundtrip Engineering
  - Changes in model reflected in code
  - Changes in code reflected in model
  - Good for structure & static stuff
    - Java/C++/C# Class - UML class
    - Method Stubs, Attributes
- Growing support for dynamic stuff
  - Sequence diagrams
- Partial support for code generators
  - Invoked called from the tool
- Support for Design Patterns
- Strong reverse engineering support
- String refactoring support

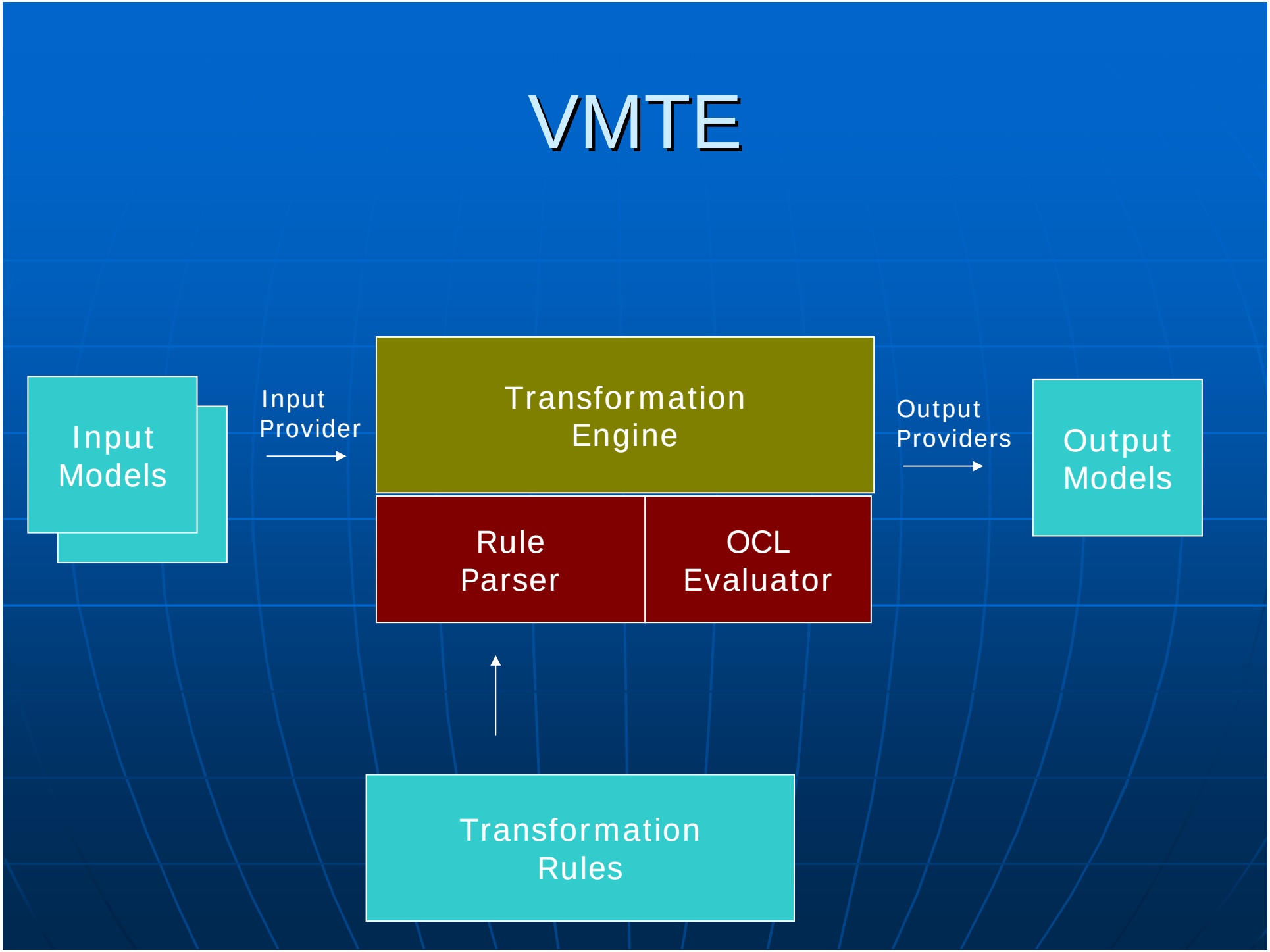
IDEs & Modeling Tools / 2

- Roundtrip Engineering
  - Not fully compatible with MDA idea
  - Problem: existing Frameworks, existing code
  - Backward engineering of code to models may conflict with UML profiles
- Different degree of support in IDEs and UML Tools
  - Invocation of code generator
  - Full-Roundtrip Engineering

MDA based Code Generators

- Models as Input
- Generates Code from these Models
- UML Profiles guide code generation
- Examples
  - AndroMDA
    - Small framework for generation of EJBs, Hibernate from UML class diagrams
    - Extensible with cartridges
  - VMTE
    - Versatile Model Transformation Engine
      - Transforms one input model to one output model
      - Not restricted to UML input!

VMTE


Input Models
Input Provider
Transformation Engine
Rule Parser
OCL Evaluator
Output Providers
Output Models
Transformation Rules

VMTE / 1

- Input Providers parse input model and feed this input in tree form to the transformation engine
- Transformation rules are parsed by transformation engine and applied on previous fed input
- Transformation rules create elements of the output model

VMTE – Transformation Language

- Same syntax for each pair of input/output protviders
- Abstract syntax for rules:

```
rule Rulename (Type t)
   [where OCL-Expression]
create TargetType (ArgList) [as TargetName] [to
   OCLExpr]
   [with alias1=OCLExpr, alias2=OCLExpr, …]
   [hide varName1, …]
   [code template]
   [applyrules OCLExpr1, … [rule Rulename.Childname]
[create …]
;
```

VMTE – Rule Language / 1

- Each rule starts with keyword rule
  - Followed by a name
  - Dots (.) in a rulename denote lower hierarchies (= allow child rules)
  - Argumentlists after rule support that input provider feeds a model element into rule evaluation
    - If model element does not type-match rule is skipped
- Example: rule A (Class myClass)
  - Tries to evaluate the rules with all Class elements of a model (sequentially, one class after the other)

VMTE – Rule Language / 2

- (optional) Where clause constrains rule evaluation based on an OCL-Expression
  - This expression may refer to the named rule argument or to previously used arguments and aliases in hierarchically higher-level rules, or already constructed target model elements
    - Whatever is accessible via OCL-Expressions
- Example:
  - where not myClass.name.startsWith("AX")

VMTE – Rule Language / 3

- "create" blocks (many are allowed)
  - Support creation of target model elements
    - "as" targetname modifies name used (only within the rules)
    - "to" adds the created model element to another model element
  - Example:
    - create CompilationUnit(myClass.name) as cu
      - Creates a compilation-unit = compilable file with name myClass.name (eg. Java) and stores it under the VMTE name "cu"
    - create Class(myClass.name, "public") to cu
      - Creates a class and stores it in the above generated compilation unit

VMTE – Rule Language / 4

- with clauses
  - Define aliases/shortcuts for common OCL-Expressions in create blocks
- Hide clauses
  - Removes argument names for the following code template or rule forwarding statement in the create block

VMTE – Rule Language / 5

- It is possible to define every target statement in a fine-grained way until a per-expression level
  - Problem: may become tedious if you have to define every piece of a statement with its one create statement
- Solution: Code templates
  - Text fragments that are inserted into the generated code
    - If output provider supports parsing this generated code can be converted in syntax tree
    - Otherwise just the text is inserted
    - Within this text VMTE variable names, aliases (with) are replaced by the appropriate value
      - Problem if variable names use common names/characters
        - Wrong text parts may be replaced
      - VMTE sorts all variables by longest to shortest name
      - Hide keywords add another chance to avoids wrong replacements

VMTE – Rule Language / 6

- Example Code Templates:

```
rule X (Class myClass)
create Class(myClass.name, "public")
with ClassName=myClass.Name
#{
  public ClassName()
  {
     Console.out.WriteLine();
  }
#}
```

VMTE – Rule Language / 7

- At the end of a create block an applyrule statement may be provided
  - Starts invocation of other (nested=child) rules at this place

VMTE

- Supports arbitrary input & output models via input & output providers
  - Currently implemented:
    - UML/XMI input provider
    - CSharpOutputProvider
      - partially
    - JavaOutputProvider
      - Partially, other stuff must be done with code templates
  - In Development
    - Java & C# InputProvider
      - detection of design patterns
      - Reverse engineering
    - UML OutputProvider

VMTE

- Very small rule language
  - Can be used for many different tasks
  - Code templates avoid writing expressions up to every detail
    - Become tedious
- Different input & output providers possible

Architectural IDEs

- IDE with mature MDA Support
- UML and MDA modeling well integrated
  - Tool enforces UML profiles
- Easily extensible
  - Support for additional UML profiles
- Support for multiple target platforms
- Model Verification support
- Example
  - ArcStyler (Interactive Objects)

Architectural IDE – ArcStyler / 1

- Pluggable Cartridges
  - Rules for model transformation
  - Rules for model verification
  - CARAT (CARtridge ArchiTecture)
  - „MDA-Cartridge Source Forge“
- Cartridge Engine
  - Services used by cartridges
  - Cartridge uses JPython as implementation language
- Example
  - UML to code
  - Abstract business model to J2EE
  - Testing code

Architectural IDE – ArcStyler / 2

- Supports modeling/generation multiple variants for
  - Middleware
    - EJB, J2EE, .NET, WebServices
  - User Interface
    - JSP, ASP.NET
  - Cartridges
- Generation of checking code for OCL
- Integration with other products
  - Eclipse, Visual Studio .NET, Apache

MDA Applicability / 1

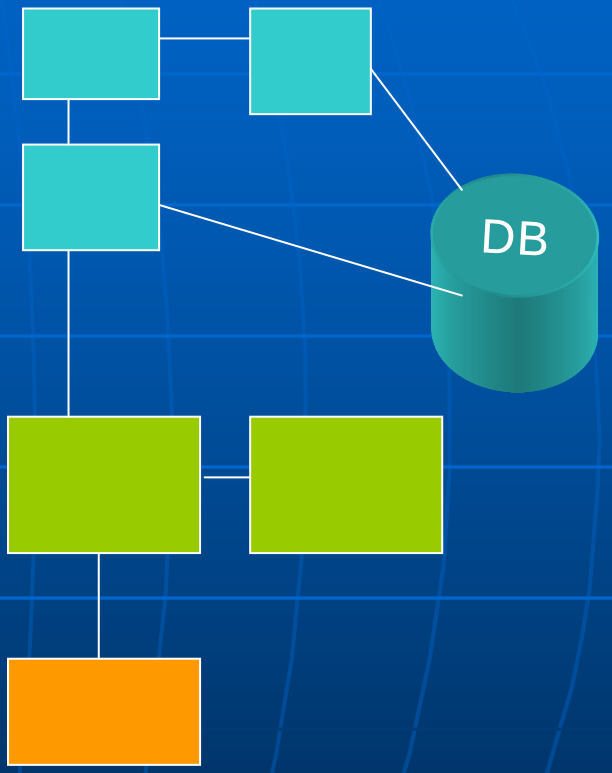
- MDA good for
  - Persistent objects
  - Generation of Datamodels
  - Those scenarios where Models are expressive enough
    - Class diagrams are well understood
  - Generation of StateMachines
  - Generation of (Design) Patterns
    - Many Patterns already supported by IDEs
  - Generation of constraint checks

MDA Applicability / 2

- MDA not so good for
  - Dynamic stuff
  - Operations (!)
  - UML Action Semantics
    - Describes semantics of operations
      - Very high abstraction level
      - Constructs for formulating database queries
    - Requires a concrete language that conforms to UML Action Semantics
      - Loss of portability

MDA Applicability / 3

- Realistic Scenario
- Generation of DataModel/Persistent objects with MDA tool
- Implementation of other Stuff manually
- Manual adaptation of PSM manually



Next Steps

- Automatic Generation of methods
  - Based on OCL
    - Query methods, Constructors
  - Based on UML Action Semantics
- Automatic Generation of database Queries
- Integrated Model Checking
  - Deadlock detection
    - Statemachines
    - Theory based on Hoare's Concurrent sequential processes (CSP)

MDA Praktikum / Master‘s Thesis

- Contact me

- MDA, UML, MOF, OCL
- MDA & Distributed & Mobile Collaboration(DMC)
- Eclipse & IDE Extensions
- Software Components
- Service Oriented Computing

Summary

- MDA – A new Hype
  - Addresses Productivity, Quality, Longevity
- Construction of Applications from Models
  - Or parts of applications
  - Requires complete/expressive models
- Bunch of new Technologies/Paradigms
  - UML/MOF,XMI,OCL
- MDA Tools are already here
  - Better tools will come in the next years
  - VMTE – versatile model transformation engine
- MDA not the solution to all and every software engineering problem
  - But solves many problems continously solved by many people