An introduction to MDR: The Model Driven approach

Frédéric Fondement

Software Engineering Lab
Swiss Federal Institute of Technology
Lausanne
Switzerland
December 2, 2003
Contents

- About the MOF
- About JMI
- The MDR tool
- Demo
- Outlook
Contents

- About the MOF
  - Generalities
  - An overview
  - Layers
  - XMI
- About JMI
- The MDR tool
- Demo
- Outlook
MOF: Generalities

- An object is « instance of » (the UML definition of) a class
- A database record is « instance of » a table schema
- Class or table schema are concepts described in a **metamodel**
  - A class is composed of attributes and operations and have a name
  - An attribute belongs to a class, have a name, a type and a multiplicity
  - ...
- A metamodel is represented as a (meta-)class model
  - Metaclasses class, attribute, operation...
- But where is described a metaclass ?
  - In a meta metamodel !
  - Should we continue (meta meta metamodel...) ?
  - Are there different kinds of meta metamodels ?
MOF : Generalities

- MOF (Meta Object Facility) is THE meta metamodel
  - The goal is to describe real world
  - Real world abstracted in a model
  - Different kinds of model
  - One way to describe a metamodel is enough!
- It is self-described!
  - The meta meta metamodel is the MOF meta metamodel
- Standard from the OMG
MOF: Layers
Let’s take a simple example of metamodel

```
NamedElement
  name: string

TypedElement
  *
  typedElement

Column
  isPrimaryKey: boolean

Table
  table
  column

DataBase
  database
  table

0..1 A_table_dataBase

0..1 A_column_table

DataType
```

Diagram:

- Element
- NamedElement
- TypedElement
- Column
- Table
- DataBase
- Column
- Table
- Database
MOF: Layers

The 4 layers architecture (here missing M0 = the real world; imagine your favorite team as a winner!)

The diagram shows the model elements and their relationships:

- **ModelElement**: name : string
- **Table**: name : string
- **DataBase**: A_table_dataBase
- **Column**: isPrimaryKey : boolean
- **TypedElement**: A_type_typedElement

The diagram also includes a table for managing teams and matches:

<table>
<thead>
<tr>
<th>Teams</th>
<th>PK,FK1</th>
<th>PK,FK2</th>
<th>PK</th>
<th>PK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Team1</td>
<td>Team2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Place</td>
<td>Date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score1</td>
<td>Score2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MOF: Layers

M3 objects...

M3

MOF

Instance Of

M2

M1

Teams

PK name

PK,FK1 Team1

PK,FK2 Team2

Place Date

Score1 Score2
MOF : Layers

M3 objects... represent a M2 model!

MOF

M3

Instance Of

Meta Objects

M2

M1

© F. Fondement, EPFL-IC-LGL - slide 9 - 4/20/2004
MOF : Layers

*M<X>* objects… represent a *M<X-1>* model!

MOF

Instance Of Meta Objects

M3

M2

M1
MOF : Model interchange (XMI)

- Model interchange is standardized
- Should take into account models of any kinds (of any metamodel)
- XMI is XML => it needs a schema
- Schema is given by the M(X+1) level!
  - Tools generate a DTD from a metamodel
  - Tools load / store models from / to XMI
  - An “XMI model” is valid for a given metamodel
    - XMI is a language template

- Remark
  - 3 versions of XMI (1.0, 1.1, 1.2)
  - Many versions of metamodel (UML : 0.9, 1.0, 1.3, 1.4, 2.0…)
  - Tools interpret the XMI standard as they wish!
  - XMI possibilities for a same model (of a given metamodel) : Cartesian product of
    - XMI version
    - Metamodel version
    - Tool
Contents

- About MOF
- About JMI
  - A MOF mapping for Java
  - Reflective facilities
  - Generated interfaces
- The MDR tool
- Demo
- Outlook
JMI : A MOF mapping for Java

- « MOF to IDL mapping » chapter
  - Concept part of the MOF standard
  - Made to access and to manipulate the model through CORBA

- JMI is all the same, but for Java
  - Just an interface definition!
  - Provides XMI facilities

Metamodel

JMI Specific Interfaces
JMI: Reflective facilities

As defined in the MOF, it is possible to

- Access the metatype of an object
- Asks a metatype for each one of its instances
- Access a feature of an object (with name of meta element)
- ...

```
RefPackage
  refClass()
  refAllLinks()
  refRemoveLink()
  refPackage()
  refAssociation()
  refImmediatePackage()
  refMofId()
  refVerifyConstraints()
  refInvokeOperation()
  ...

RefAssociation
  refMetaObject()
  refImmutablePackage()
  ...

RefObject
  refIsInstanceOf()
  refClass()
  refInviteOperation()
  ...

RefBaseObject
  refMofId()
  refVerifyConstraints()
  refInvokeOperation()
  ...

RefFeatured
  refSetValue()
  refGetValue()
  ...

RefClass
  refCreateInstance()
  ...
```
JMI : Generated interfaces

- It is possible to access “meta objects” here
  - “Element” interface extends RefObject, so have a RefClass
  - Can access the meta properties of an element
    - Name (any direct “DataBase” instance returns the “DataBase” string)
    - Contents (applied on any NamedElement returns the “name” meta-attribute)
    - …
JMI : Generated interfaces

- In order to create an object, you must contact its metaclass
- A metaclass is a *singleton*
JMI : Generated interfaces

public interface ATableDataBase
extends javax.jmi.reflect.RefAssociation {
    public boolean exists(Table table, DataBase dB);
    public Collection getTable(DataBase dB);
    public DataBase getDataBase(Table table);
    public boolean add(Table table, DataBase dB);
    public boolean remove(Table table, DataBase dB);
}

A meta association is a singleton
JMI: Generated interfaces

- In order to create an object, you must contact its metaclass
- A meta element is a *singleton* and provide access to its nested meta elements
- The root meta package is the entry point to access these singletons
- Need to be provided a mechanism to retrieve the root package singleton

```java
public interface DataBasePackage
    extends javax.jmi.reflect.RefPackage {
    public ElementClass getElement();
    public NamedElementClass getNamedElement();
    public DataBaseClass getDataBase();
    public TableClass getTable();
    public ATableDataBase getATableDataBase();
}
```
Contents

• About MOF
• About JMI
• The MDR tool
  • An implementation of JMI
  • Architecture
• Demo
• Outlook
MDR : An implementation of JMI

An open-source tool from Sun

- MDR provides access to the meta package we needed above
- It is able to manage any model of any (MOF) metamodel
- It can generate the JMI interfaces
- It provides an implementation for these interfaces
- It provides XMI support
  - Reader
  - Writer
  - DTD generation

MDR is a model repository
MDR : Architecture
Contents

- About JMI
- The MDR tool
- Demo
- Outlook
Contents

- About JMI
- The MDR tool
- Demo
- Outlook
  - Just describe a metamodel to build a repository
  - Manipulate your models as you manipulate objects
  - No support for profile or constraint…
  - Tricky support for operations and constraints
  - Many tools use MDR
    - The new version of Dresden OCL Toolkit
    - Poseidon
    - The model transformation languages MTL and ATL
    - …
Thank you!

- Any question?