

An Introduction to JMI

Vincent@Xcesium

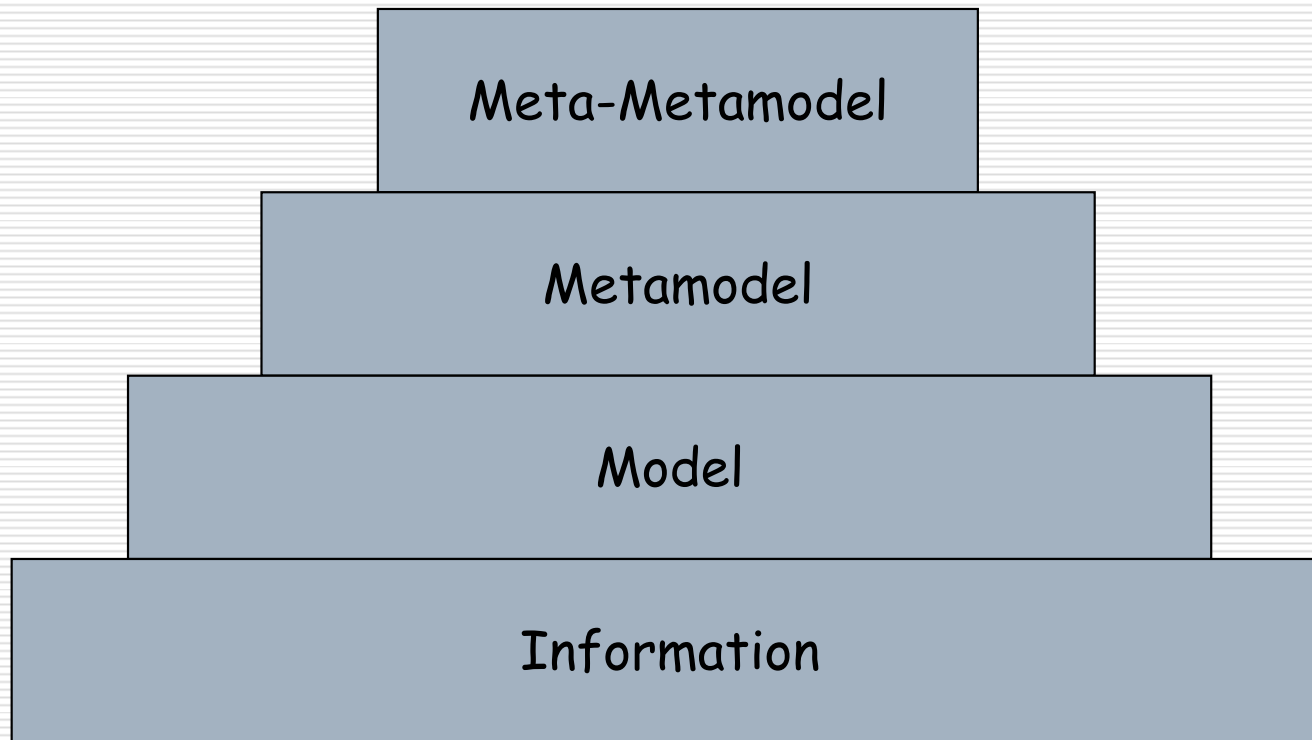
Agenda

- JMI Overview
 - MOF Overview
 - MOF-Java Mapping
 - MOF Reflection API
 - JMI Examples
 - Meta Modeling
-

JMI Overview

- JMI is the Java rendition of the MOF
 - Metadata service for the Java platform
 - A metadata framework that provides a common Java programming model for accessing metadata
 - An integration and interoperability framework for Java tools and applications
 - Integration with OMG modeling and metadata architecture
-

MOF Overview — Four Layers



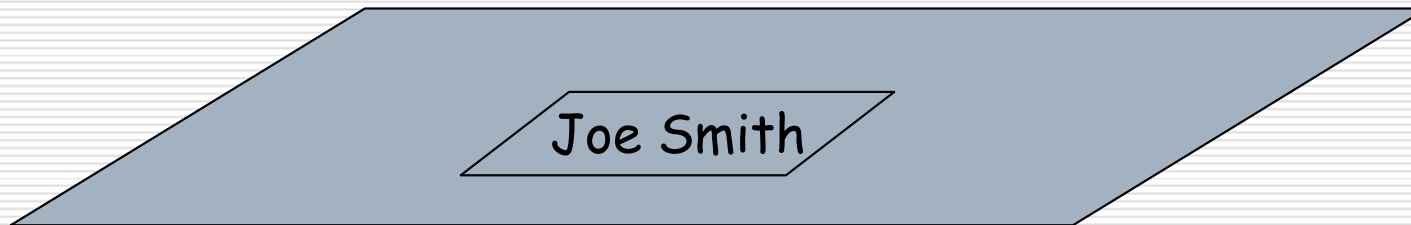
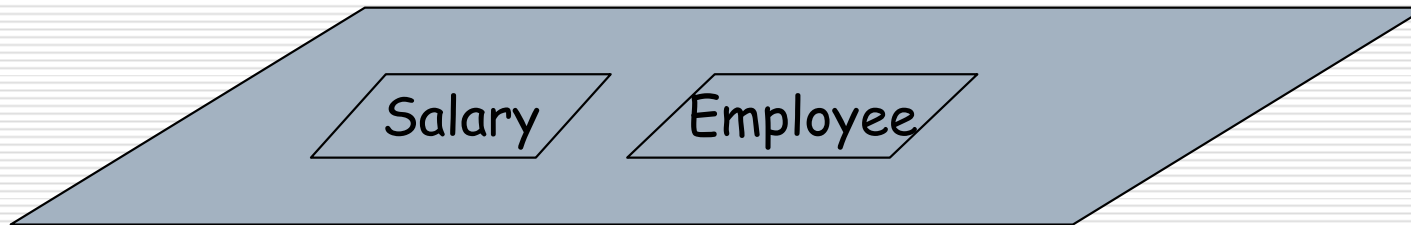
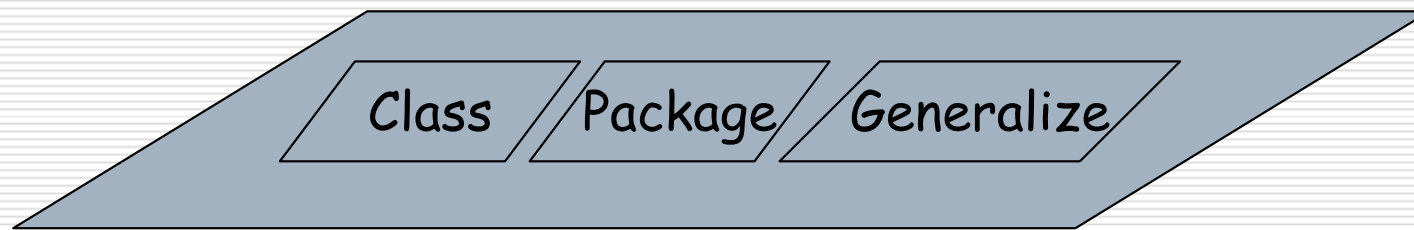
MOF Overview — Modeling Perspective

Concept/Knowledge/Domain

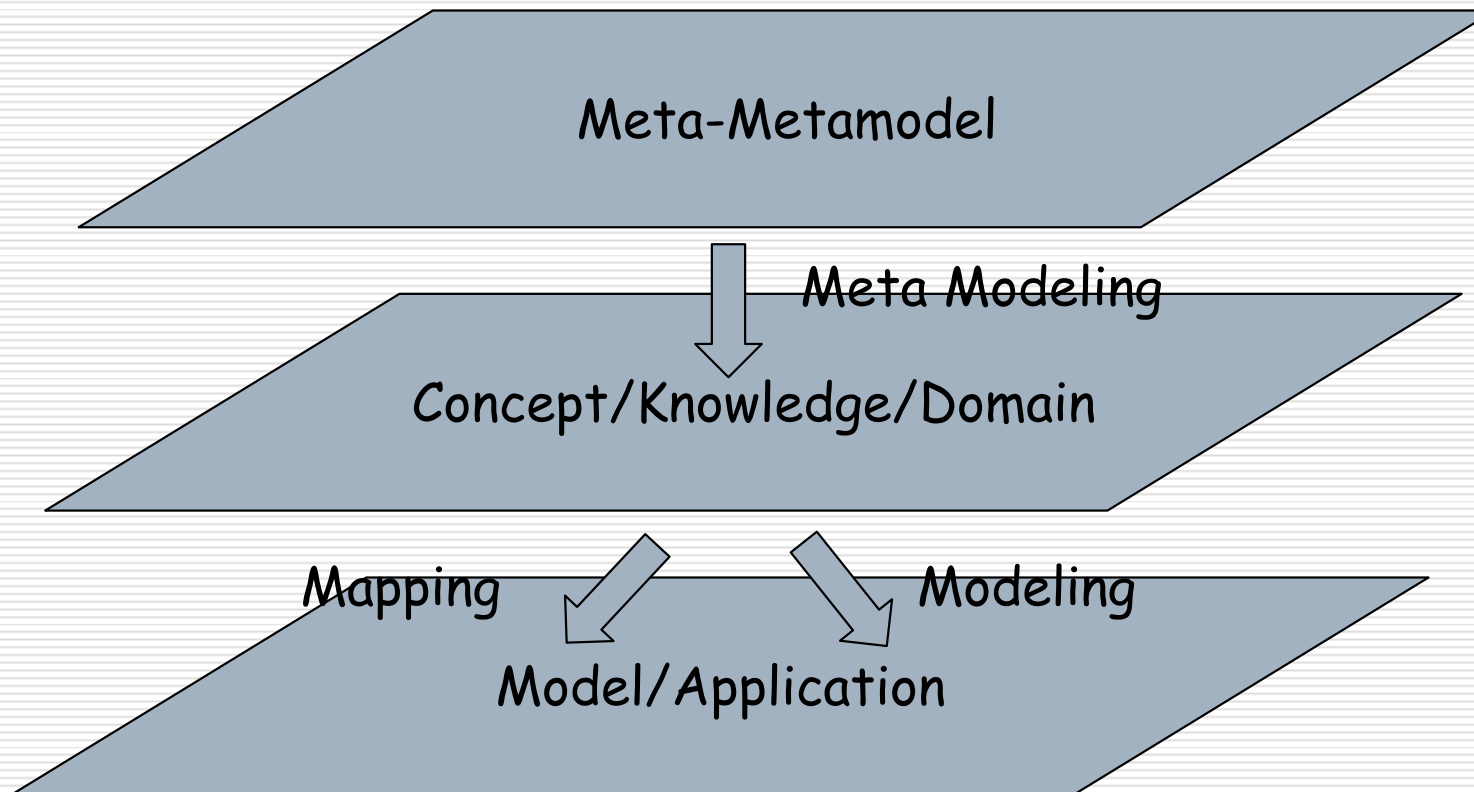
Model/Application

Information/Runtime

MOF Overview — Modeling Perspective



MOF Overview — Modeling Perspective

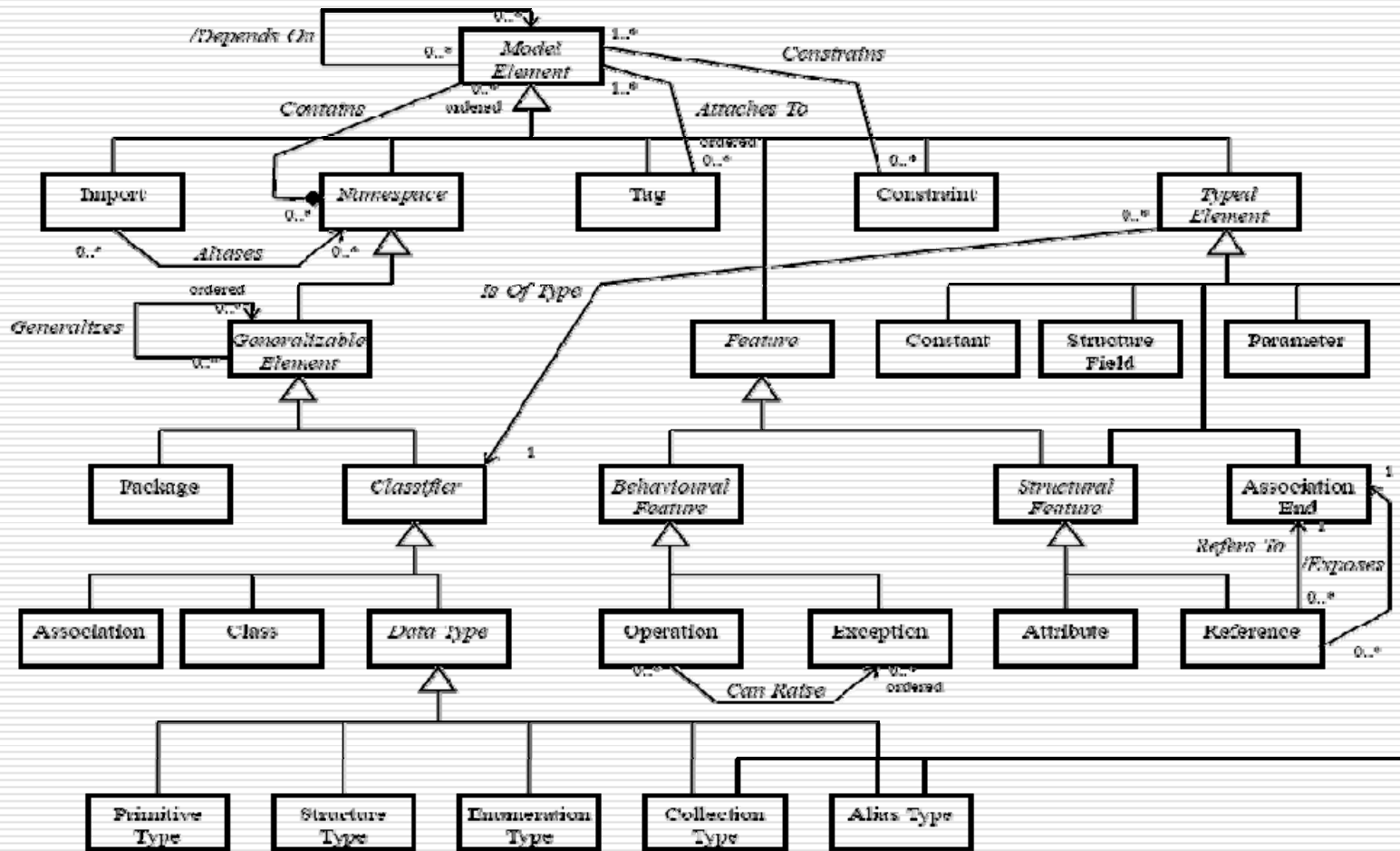


MOF Overview

- Knowledge Modeling
 - MOF Model
 - MOF Mapping
 - MOF Reflection API
 - MOF Metaobjects
 - MOF Transform
 - MOF 2 Query/View/Transform
-

MOF Overview — MOF Model

Figure 3-1 The MOF Model Package



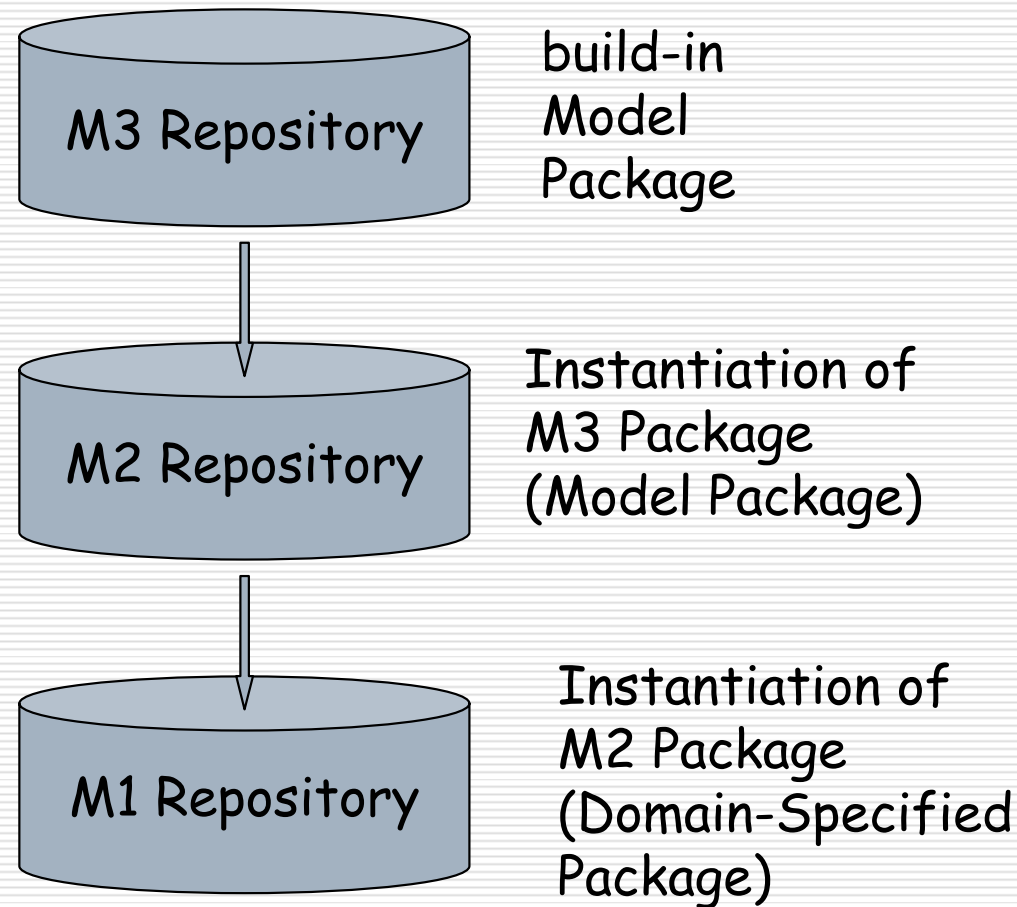
MOF-Java Mapping—Metaobject Type

- Package Object
 - Class Proxy Object
 - Instance Object
 - Association Object
 - Data Type Object
-

MOF-Java Mapping — Package Object

- A package object is little more than a “directory” of operations that give access to a collection of metaobjects described by a metamodel
 - “Container” of other metaobjects
 - Metadata Repository
-

MOF-Java Mapping — Package Object



MOF-Java Mapping — Class Proxy Object

- ❑ It is a factory object for producing instance objects within the Package extent
 - ❑ It is the intrinsic container for instance objects
 - ❑ It holds the state of any classifier-scoped attributes for the class
 - ❑ It provides operations corresponding to classifier-scoped operations
-

MOF-Java Mapping — Instance Object

- ❑ Operations to access and update the instance-scoped and classifier-scoped attributes
 - ❑ Operations corresponding to instance-scoped and classifier-scoped operations
 - ❑ Operations to access and update associations via reference
-

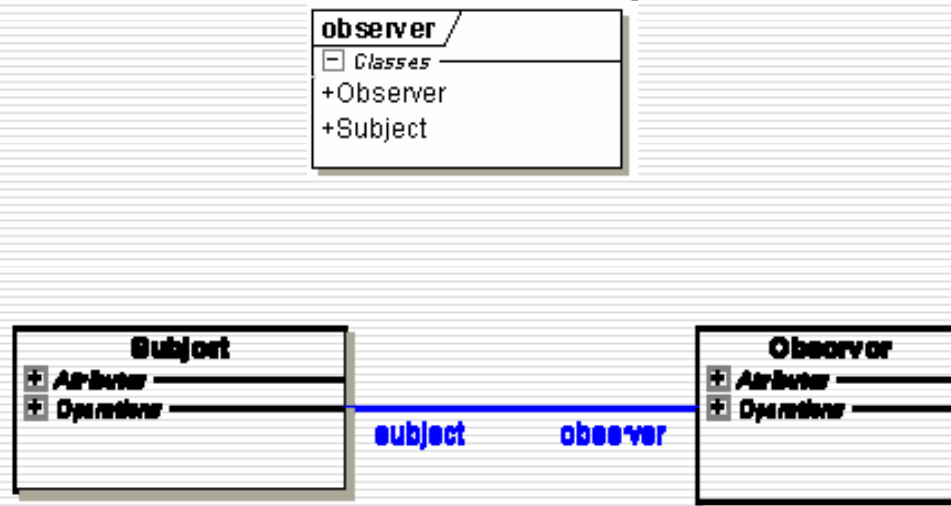
MOF-Java Mapping — Association Object

- ❑ Operations for querying the link set
 - ❑ Operations for adding, modifying and removing links from the set
 - ❑ An operation that returns the entire link set
-

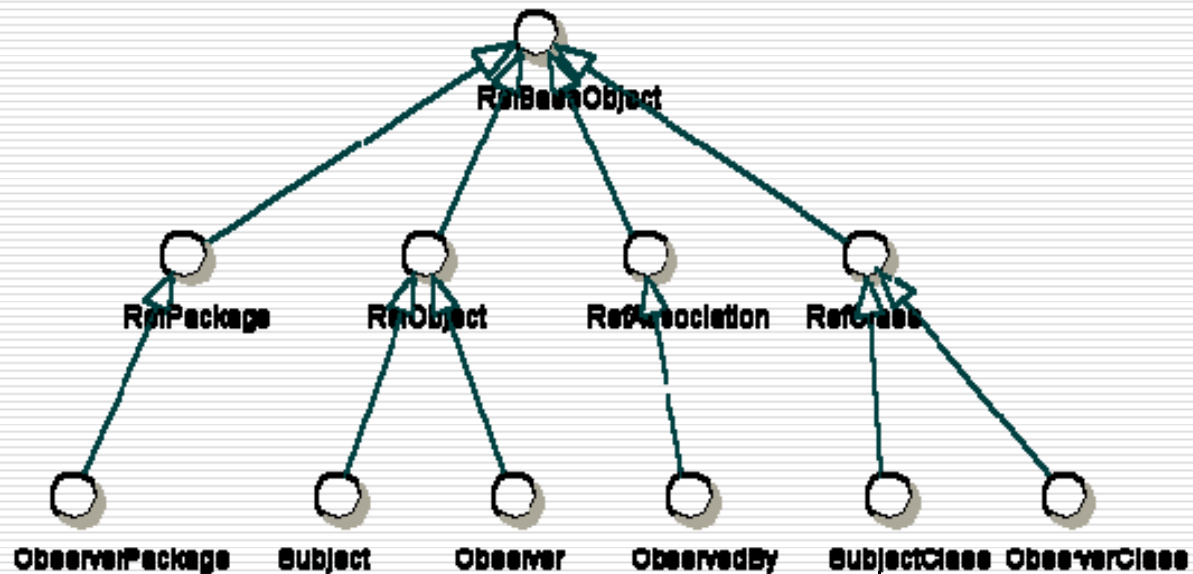
MOF-Java Mapping — Data Type

- Primary Types
 - Enumeration
 - Structure
-

MOF-Java Mapping — Metaobject Interface Hierarchy



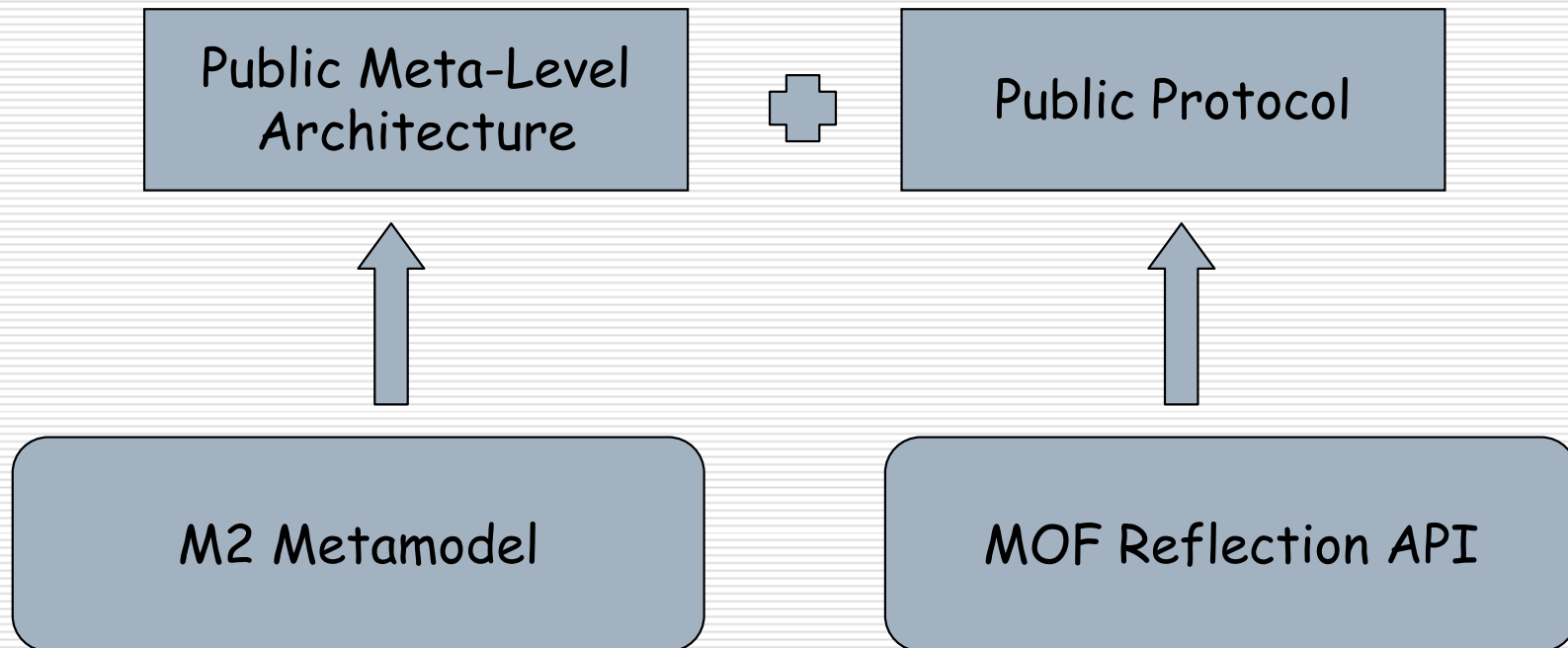
MOF-Java Mapping — Metaobject Interface Hierarchy



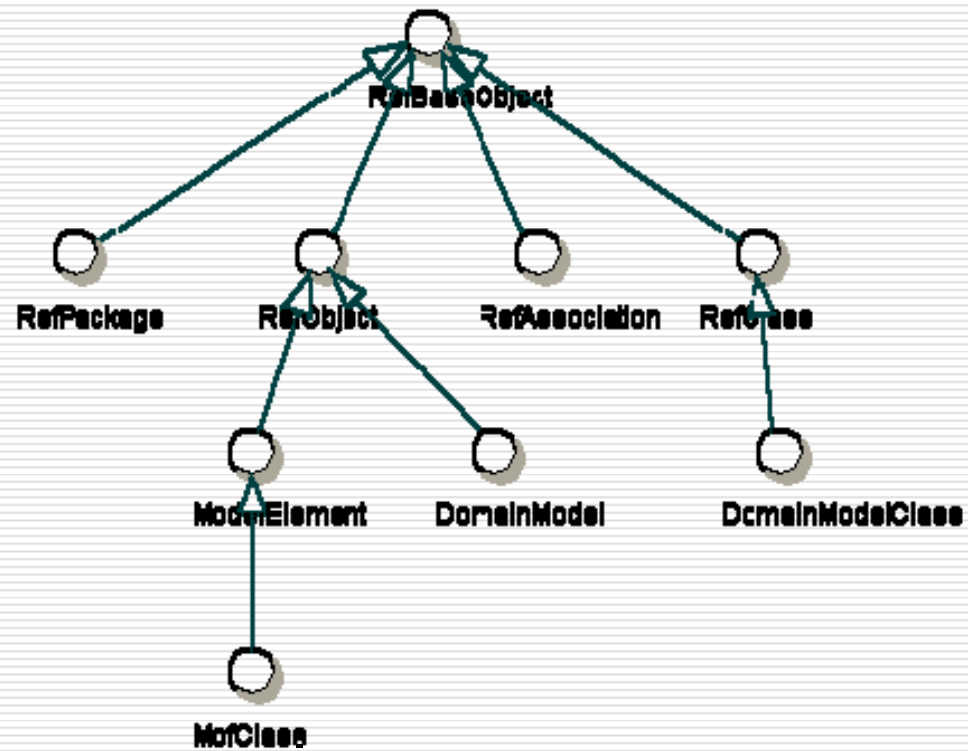
MOF Reflection API

- ❑ Create, update, access, navigate and invoke operations on class proxy objects
 - ❑ Query and update links using association objects
 - ❑ Navigate MOF package structure
 - ❑ Find an object's metaobject
 - ❑ Find an object's container(s) and enclosing package(s)
 - ❑ Test for object identity
 - ❑ Delete a object
-

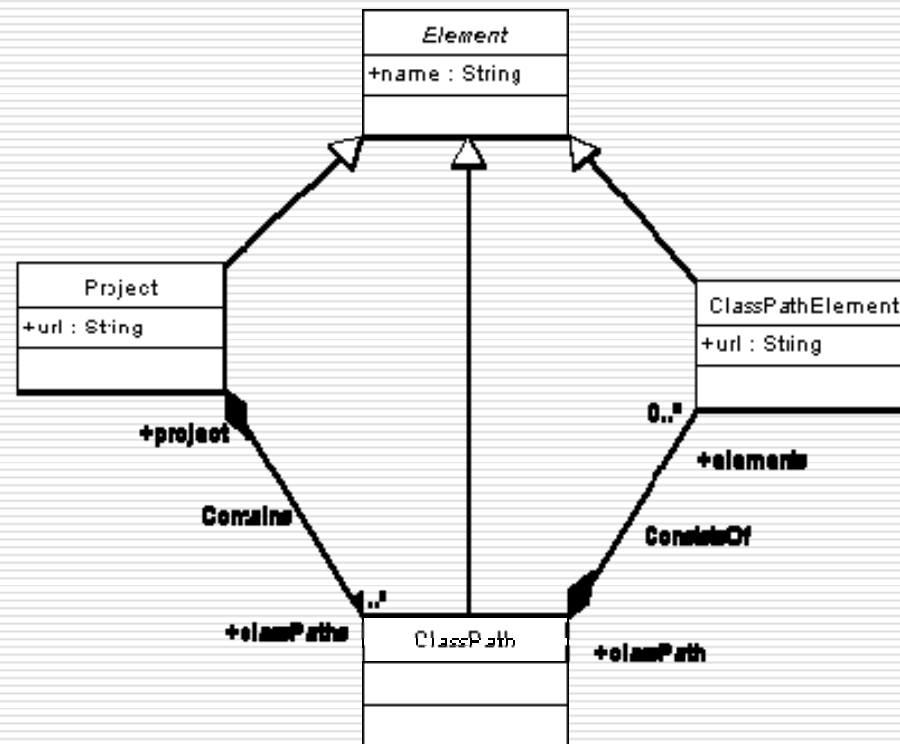
MOF Reflection API



MOF Reflection API



JMI Examples — Projects (from NetBeans.org)



JMI Examples — Create Repository (1)

Create M2 Meta Repository (Model Factory):

```
RefPackageFactory factory = new  
    RefPackageFactory();
```

```
RefPackage m2 = factory.createM2();
```

M2 is an instance of
`javax.jmi.model.ModelPackage`

JMI Examples — Create Repository (2)

Load or Create M2 Model :

```
XmiReader reader = new XmiReaderImpl();  
reader.read("file:project.xml", m2);
```

Or

```
RefClass packageClass = m2.refClass("Package");  
RefObject p1 =  
packageClass.refCreateInstance(Arrays.asList(new  
Object[] {...}));
```

.....

JMI Examples —— Create Repository (3)

Find domain package metaobject

```
RefObject repoMetaObject = null;
Iterator itr = m2.refClass("Package").refAllOfClass().iterator();
while (itr.hasNext()) {
    RefObject obj = (RefObject) itr.next();
    if (obj.refGetValue("name").equals("Projects")){
        repoMetaObject = obj;
        break;
    }
}
```

JMI Examples — Create Repository (4)

Create M1 Repository

```
RefPackage m1 = factory.createM1(Arrays  
    .asList(new Object[] { repoMetaObject }));
```

JMI Examples — Create Repository (5)

Test the Repository

```
// get Project Class
RefClass projectClass = m1.refClass("Project");
RefObject project = projectClass.refCreateInstance(Arrays
    .asList(new Object[] { "Test Project", "d:/test_project" }));
// get ClassPath Class
RefClass classPathClass = m1.refClass("ClassPath");
RefObject classPath = classPathClass.refCreateInstance(Arrays
    .asList(new Object[] { "d:/test_project/lib/a.jar" }));
classPath.refSetValue("project", project);

// write to m1 repository
XmiWriter writer = new XmiWriterImpl();
FileOutputStream out = new FileOutputStream("create_m1_ref.xml");
writer.write(out, m1, "1.2");
out.close();
```

JMI Examples — Create Repository (6)

In create_m1_ref.xml

```
<XMI.content>  
<Project xmi.id='0' name='Test Project'  
  url='d:/test_project'>  
<Project.classPaths>  
<ClassPath xmi.id='1' name='d:/test_project/lib/a.jar'>  
</ClassPath>  
</Project.classPaths>  
</Project>  
</XMI.content>
```

JMI Examples — Using Gen API

```
// create m1 repo
ProjectsPackage m1 =
    ProjectsPackageImpl.create("file:project.xml");
Project project = m1.getProject().createProject("testProject",
    "d:/project");
ClassPath path = m1.getClassPath().createClassPath(
    "d:/test_project/lib/a.jar");
path.setProject(project);

// write to m1 repo
XmiWriter writer = new XmiWriterImpl();
FileOutputStream out = new
    FileOutputStream("create_m1_gen.xml");
writer.write(out, m1, "1.2");
out.close();
```

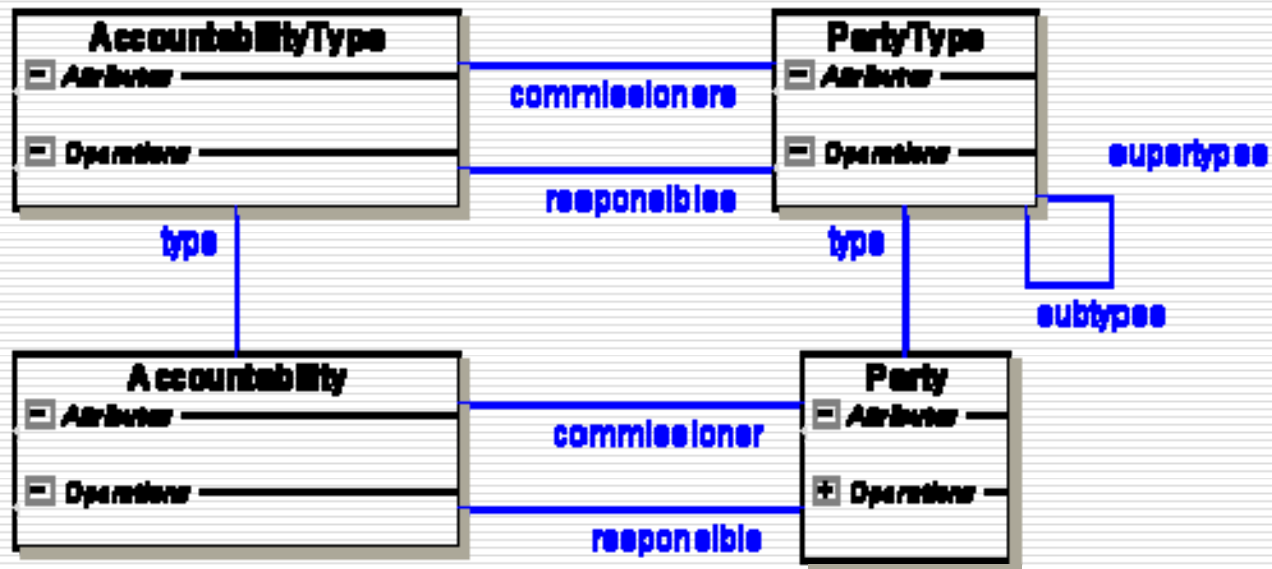
Known Uses

- ❑ JOLAP (CWM OLAP Mapping)
 - ❑ JDataMine (CWM DM Mapping)
 - ❑ J2EE Management (J2EE Metamodel)
 - ❑ IBM j2eemof
-

Meta Modeling — Accountability



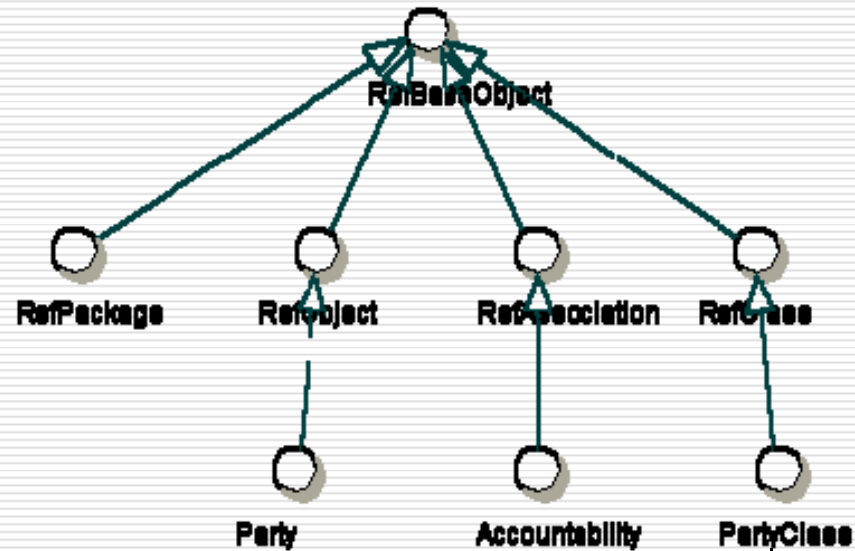
Meta Modeling — Accountability



Meta Modeling —— Accountability



Meta Modeling — Accountability



AOP Metamodel

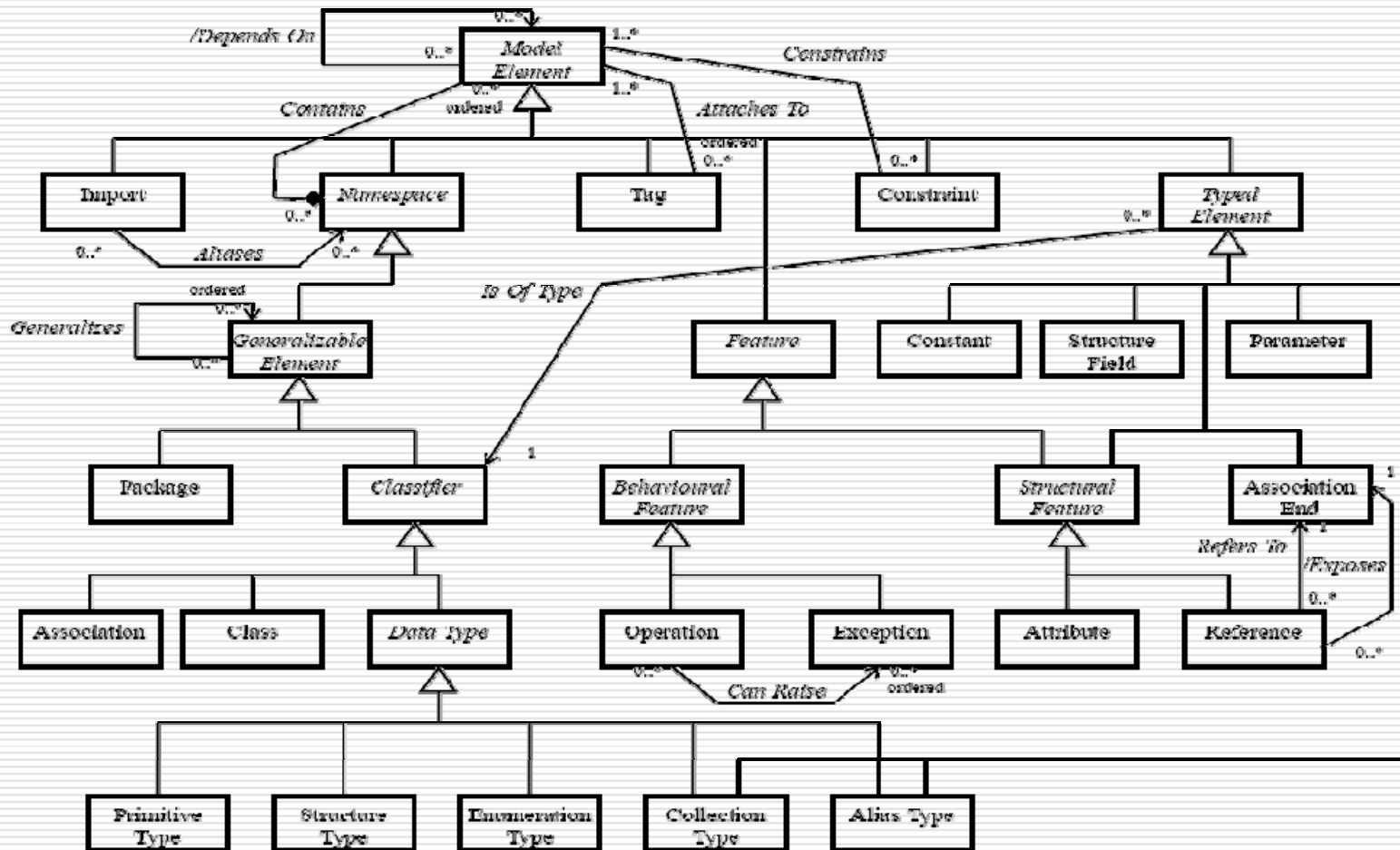
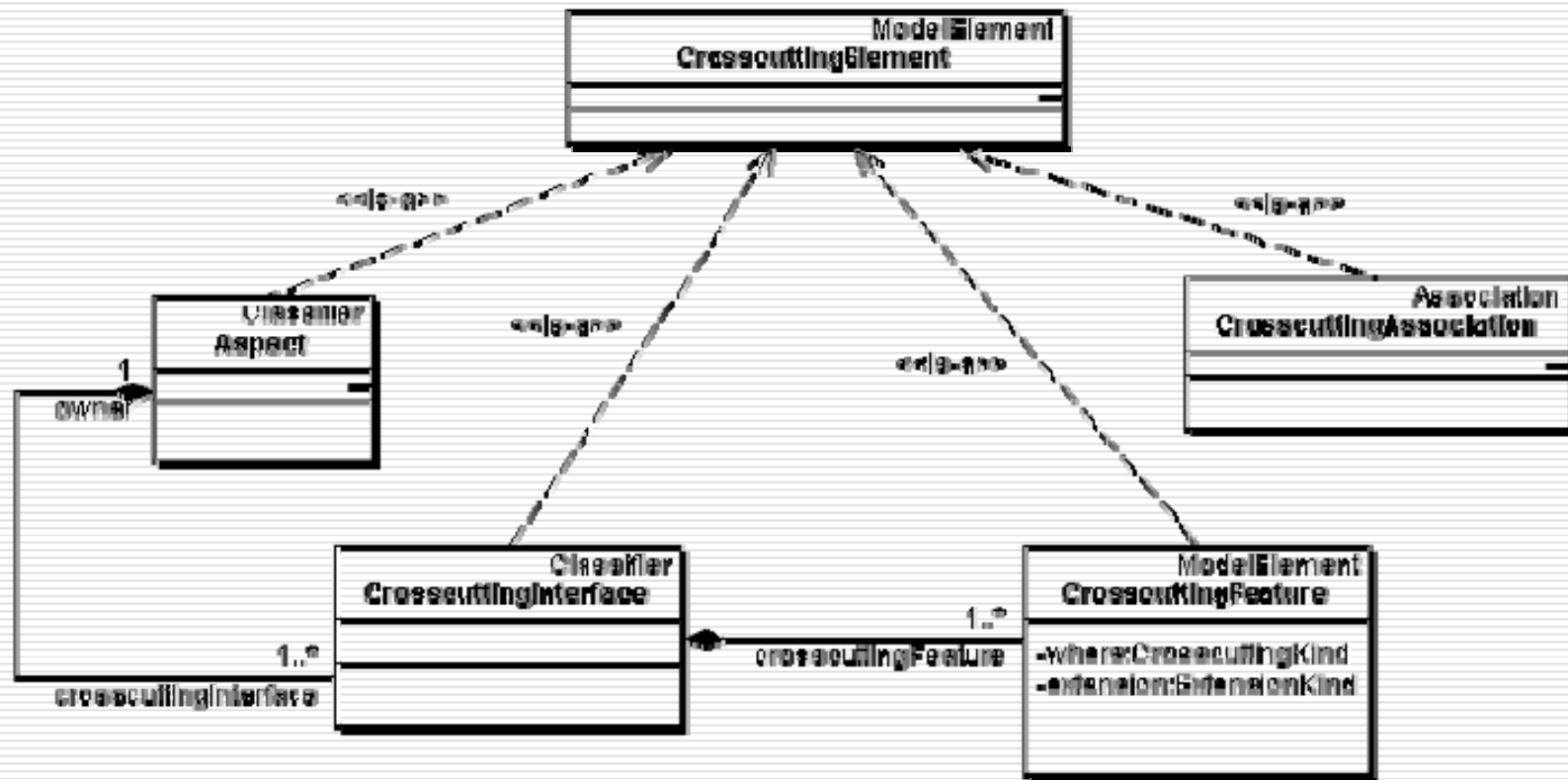


Figure 3-1 The MOF Model Package

AOP Metamodel — Chavez02



Why AOP Metamodel?

- ❑ Reflection API
- ❑ Domain-Specified AOP

Management	Warehouse Process			Warehouse Operation		
Analysis	Transformation		OLAP	Data Mining	Information Visualization	Business Nomenclature
Resource	Object Model	Relational	Record	Multidimensional		XML
Foundation	Business Information	Data Types	Expression	Keys and Indexes	Type Mapping	Software Deployment
Object Model						

Thanks

