An introduction to MOF

MetaObject Facility

pierre-alain.muller@irisa.fr

About MOF Meta-Object Facility

- **The MetaObject Facility Specification** is the foundation of OMG's industry-standard environment where models can be exported from one application, imported into another, transported across a network, stored in a repository and then retrieved, rendered into different formats (including XMI, OMG's XML-based standard format for model transmission and storage), transformed, and used to generate application code.

http://www.omg.org/mof/

Contributors

- Adaptive
- Borland
- Ceira Technologies
- Compuware
- Data Access Technologies
- DSTC
- Gentleware Intellicorp
- Hewlett-Packard
- Hyperion
- International Business Machines
- IONA
- Kinetium
- MetaMatrix
- Project Technology
- SOFTEAM
- Sun Microsystems
- Telelogic AB
- Unisys
- University of Kent
- University of York
- X-Change Technologies Group
- 88solutions

About MOF metamodels

- Metamodels provide a platform independent mechanism to specify the following
  - Structure, syntax and semantics of technology and tool frameworks
  - Programming model for any resultant metadata (using Java, IDL etc)
  - Interchange format (using XML)

MOF Conformance

- Essential MOF (EMOF)
- Complete MOF (CMOF)

- Compliant products shall support one or more technology mappings
  - MOF 2.0 XMI (ptc/04-06-11),
  - MOF 2.0 IDL,
  - MOF 2.0 JMI

MOF2 / UML2

- The MOF 2 Model builds on a subset of UML 2.0 Infrastructure which provides concepts and graphical notation for the MOF Model
- The MOF Model also includes additional capabilities defined in separate packages including support for
  - identifiers,
  - additional primitive types,
  - reflection,
  - simple extensibility through name-value pairs.

MOF Evolution

- The OMG adopted the MOF 1.1 specification in November 1997 coincident with the adoption of UML 1.1
- Since then, MOF Revision Task Forces have produced several minor revisions, the most recent being the MOF 1.4 specification, which was adopted in October 2001

Revision and evolution

- 150 formal usage and implementation issues
- Partly addressed by MOF 1.4
- Partly addressed by MOF 2.0 RFPs
  - MOF 2.0 Core,
  - MOF 2.0 IDL Mapping,
  - MOF 2.0 XMI Mapping,
  - MOF 2.0 Versioning
  - MOF 2.0 Query/View/Transformations,
  - MOF 2.0 Facility RFP

Usages of MOF

- Standard OMG metamodels and technologies
  - UML, MOF itself, CWM, SPEM, Java EJB, EDOC, EAI…
  - Various UML profiles
  - XMI, JMI

Towards MOF 2

- Reusing UML2 infrastructure
  - Using import
- Essential MOF (EMOF)
  - Simple classes with attributes and operations
  - Ensure basic and stable mapping from MOF to XML and Java

Design concerns and goals

- Ease of use in defining and extending existing and new metamodels and models of software infrastructure
- Making the MOF model itself much more modular and reusable
- The use of model refactoring to improve the reusability of models
- Ensure that MOF 2.0 is technology platform independent (J2EE, .Net, CORBA, Web Services...)

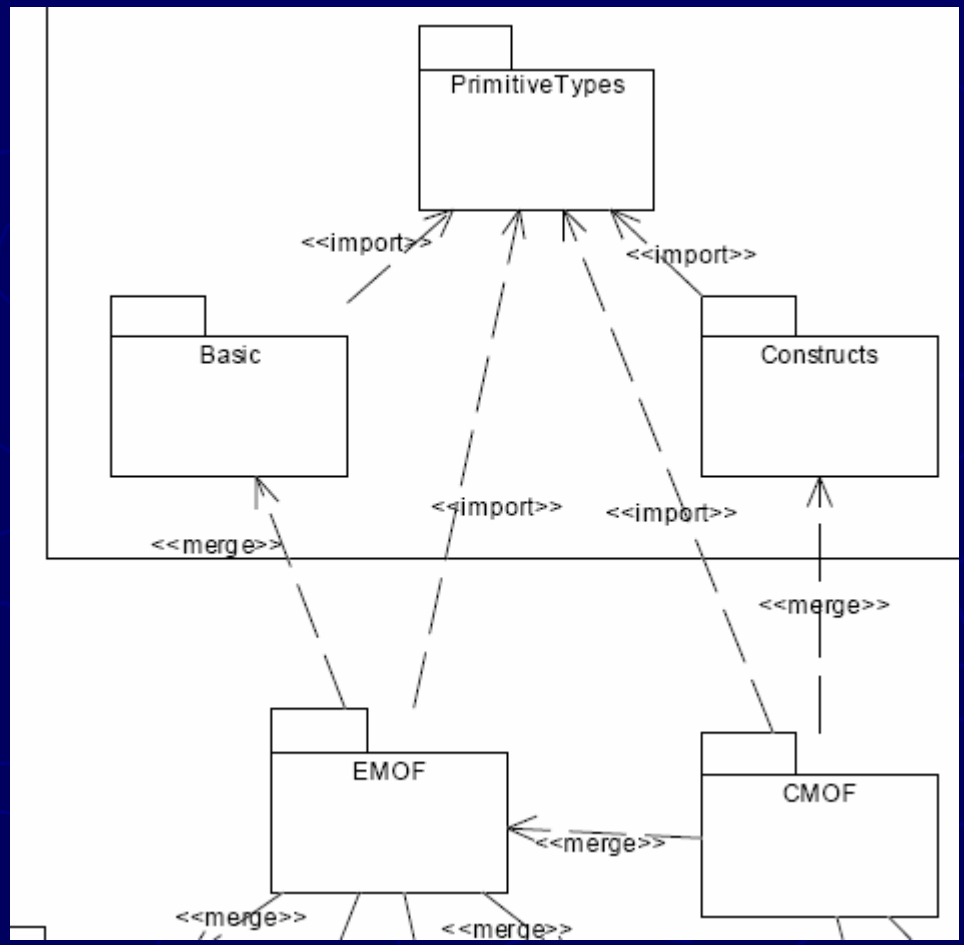
Design concerns and goals

- Orthogonality (or separation of concerns) of models and the services (utilities)
  - Metadata interchange, Reflection, Federation, Life Cycle, Versioning, Identity, Queries, etc.
- MOF 2.0 models reflection using MOF itself
- MOF 2.0 models the concept of identifier

Reuse of Common Core Packages by UML 2.0 and MOF 2.0

- Importing packages makes model elements contained in the imported package visible in the importing package
- Merging packages extends model elements in the merging package with new feature deltas from the merged package
- Standard class modeling concepts (importing, subclassing, adding new classes, associations and adding associations between existing classes) are used for MOF 2.0 extensibility

Package Import / Merge



Package Merge

- Package merging combines the features of the merged package with the merging package to define new integrated language capabilities
- After package merge, classes in the merging package contain all the features of similarly named classes in the merged package

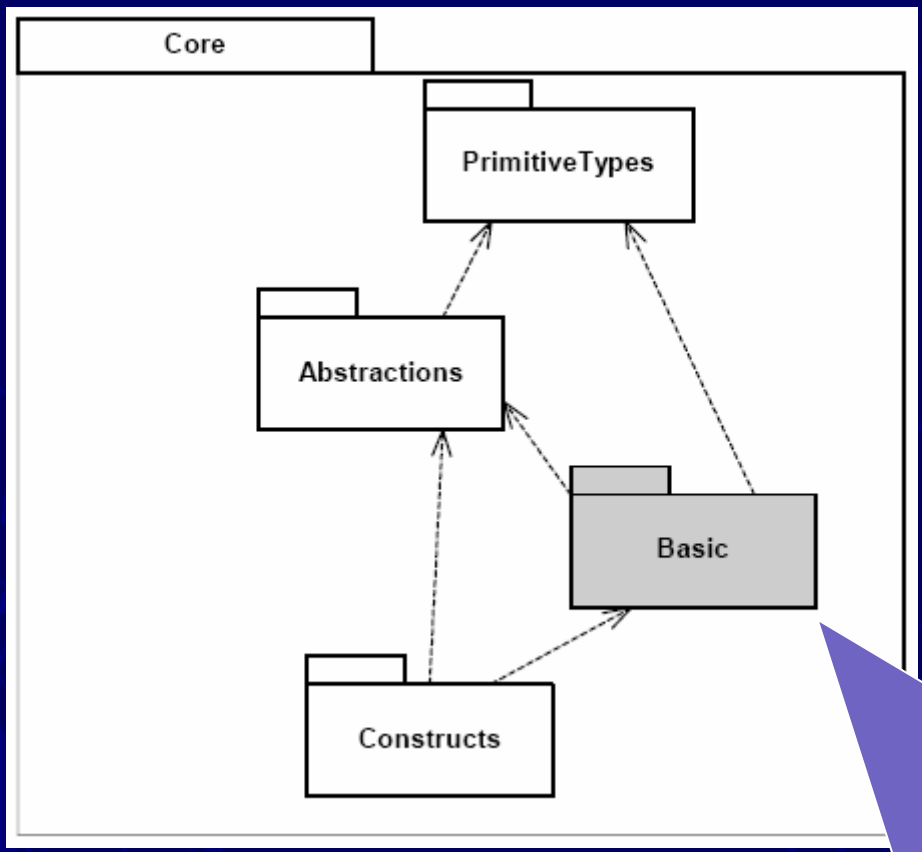
Essential MOF (EMOF)

- Subset of MOF that closely corresponds to the facilities found in OOPLs and XML
- Straightforward framework for mapping MOF models to implementations such as JMI and XMI for simple metamodels

Essential MOF (EMOF)

- EMOF merges the Basic package from UML2 and includes additional language capabilities defined in this specification
  - Reflection, Identifiers, and Extension capability packages to provide services for discovering, manipulating, identifying, and extending metadata

UML Infrastructure

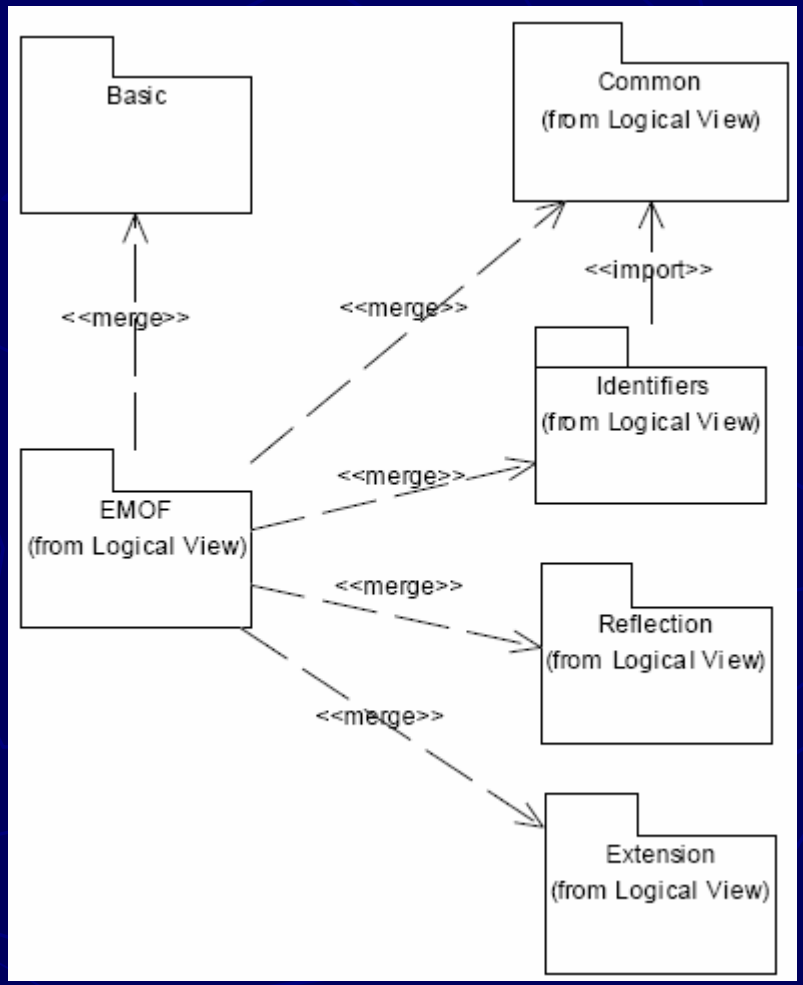


EMOF

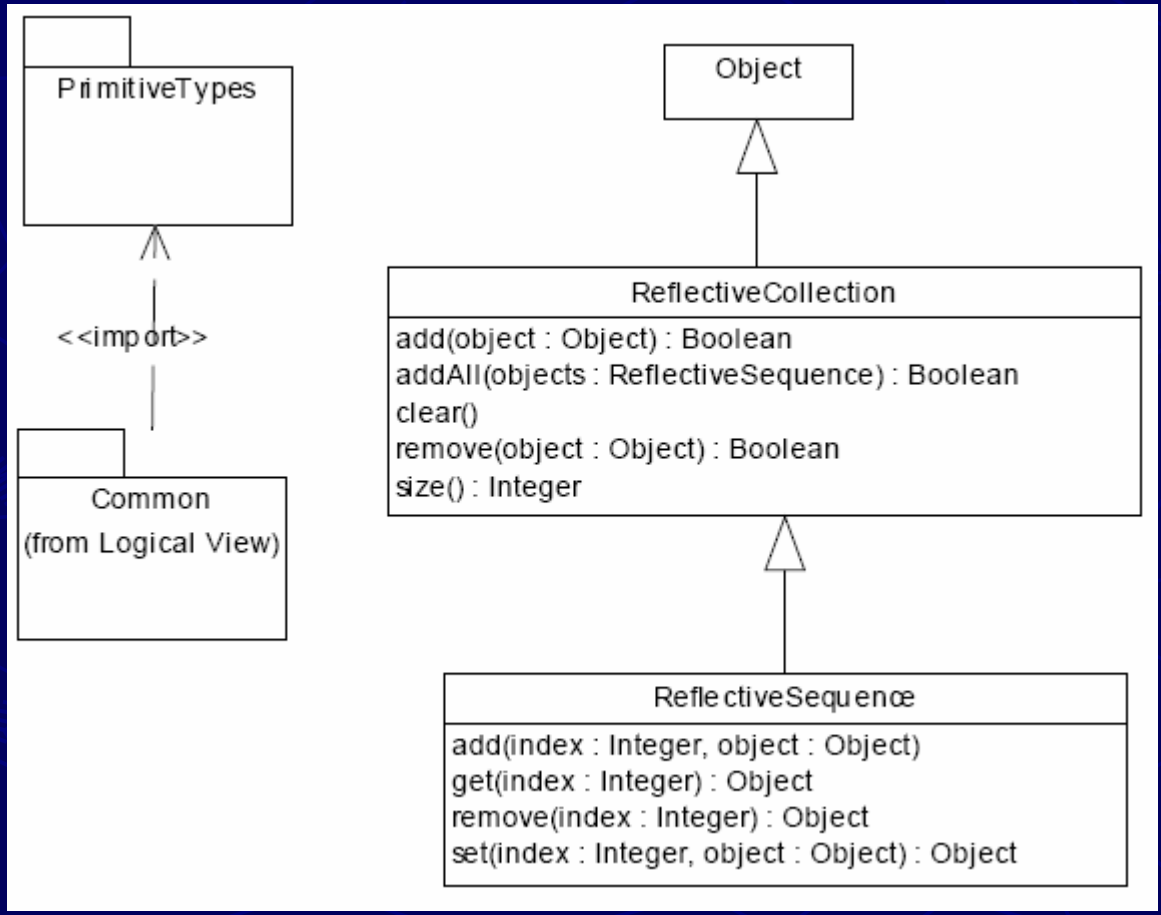
Reflection
Identifier
Extension

EMOF merges InfrastructureLibrary::Core::Basic
from UML 2.0 Infrastructure.

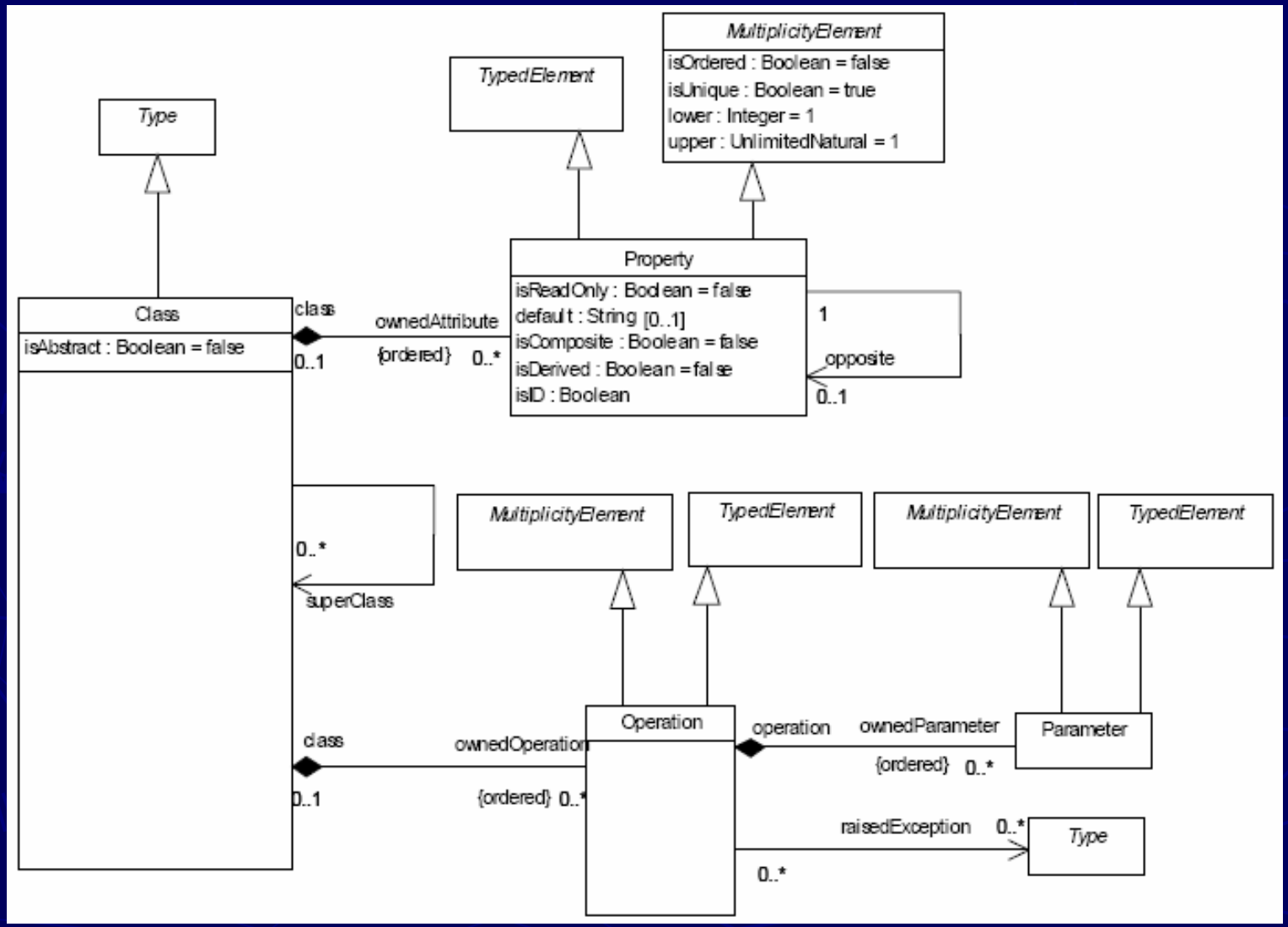
EMOF Overview



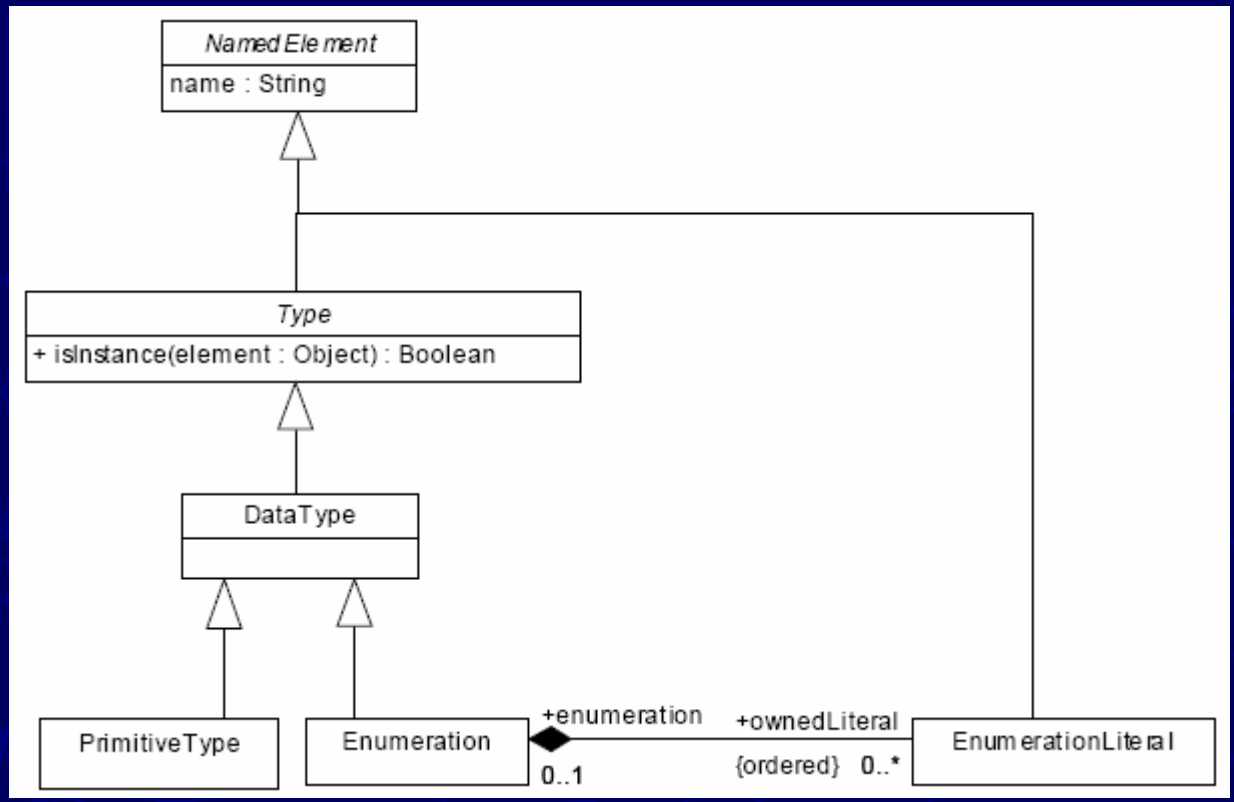
MOF Common


PrimitiveTypes
<<import>>
Common
(from Logical View)
Object
ReflectiveCollection
add(object : Object) : Boolean
addAll(objects : ReflectiveSequence) : Boolean
clear()
remove(object : Object) : Boolean
size() : Integer
ReflectiveSequence
add(index : Integer, object : Object)
get(index : Integer) : Object
remove(index : Integer) : Object
set(index : Integer, object : Object) : Object

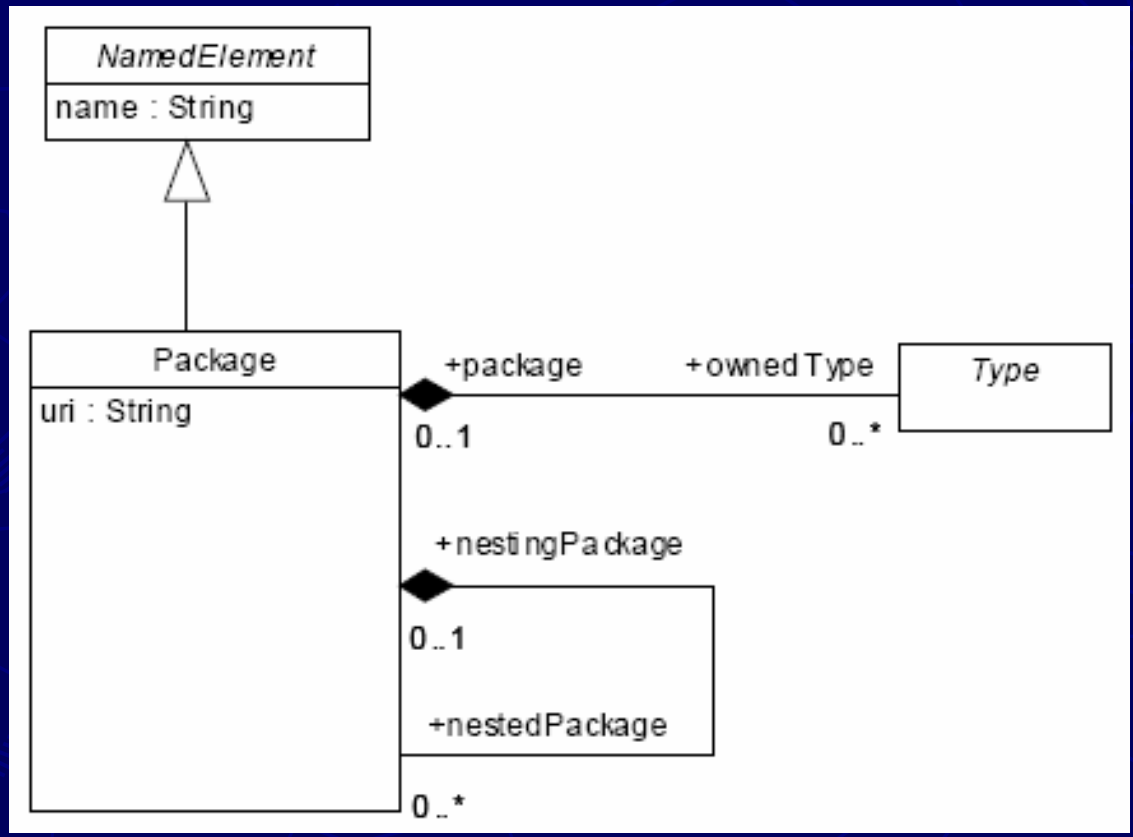
EMOF Classes



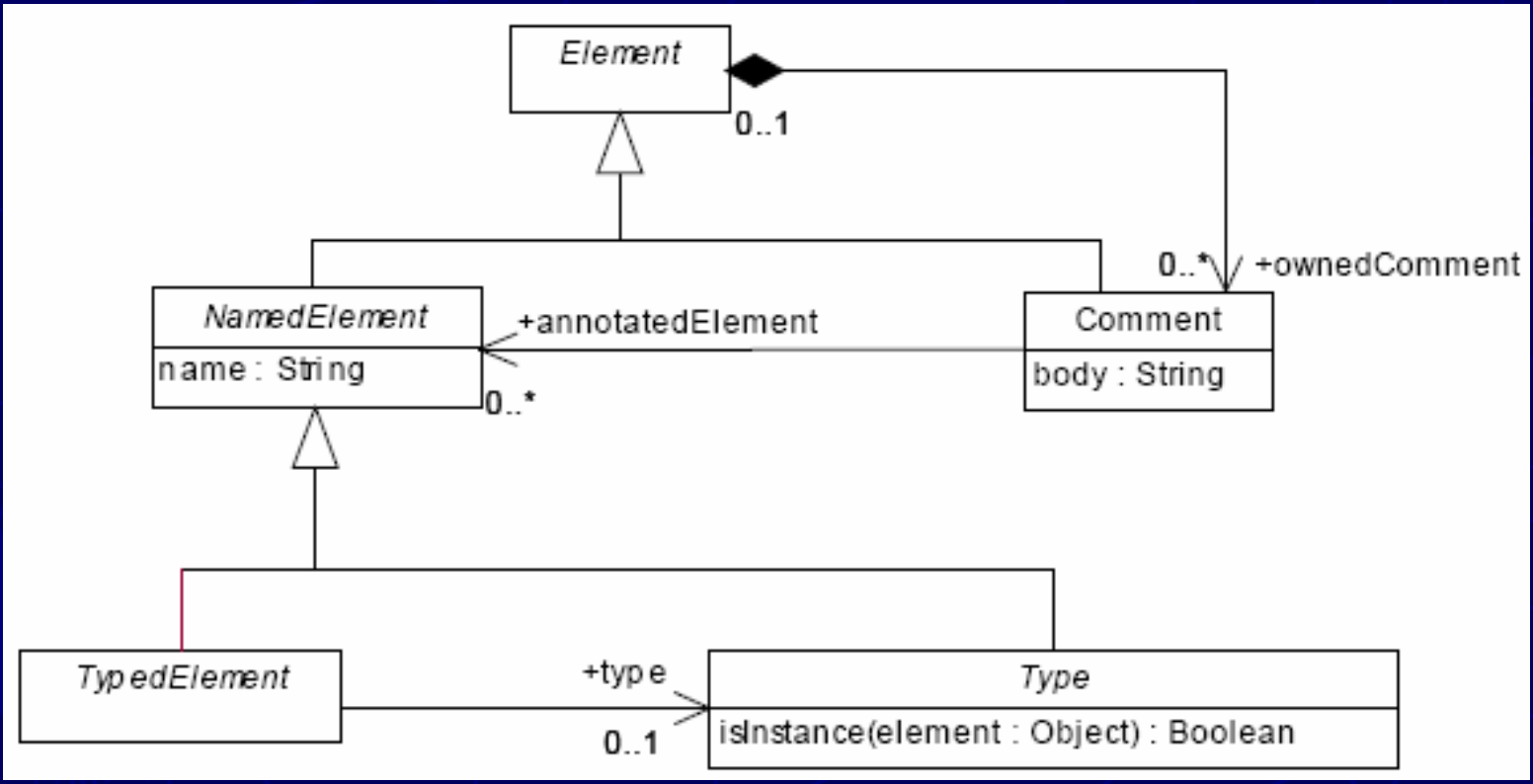
EMOF Data Types



EMOF Packages



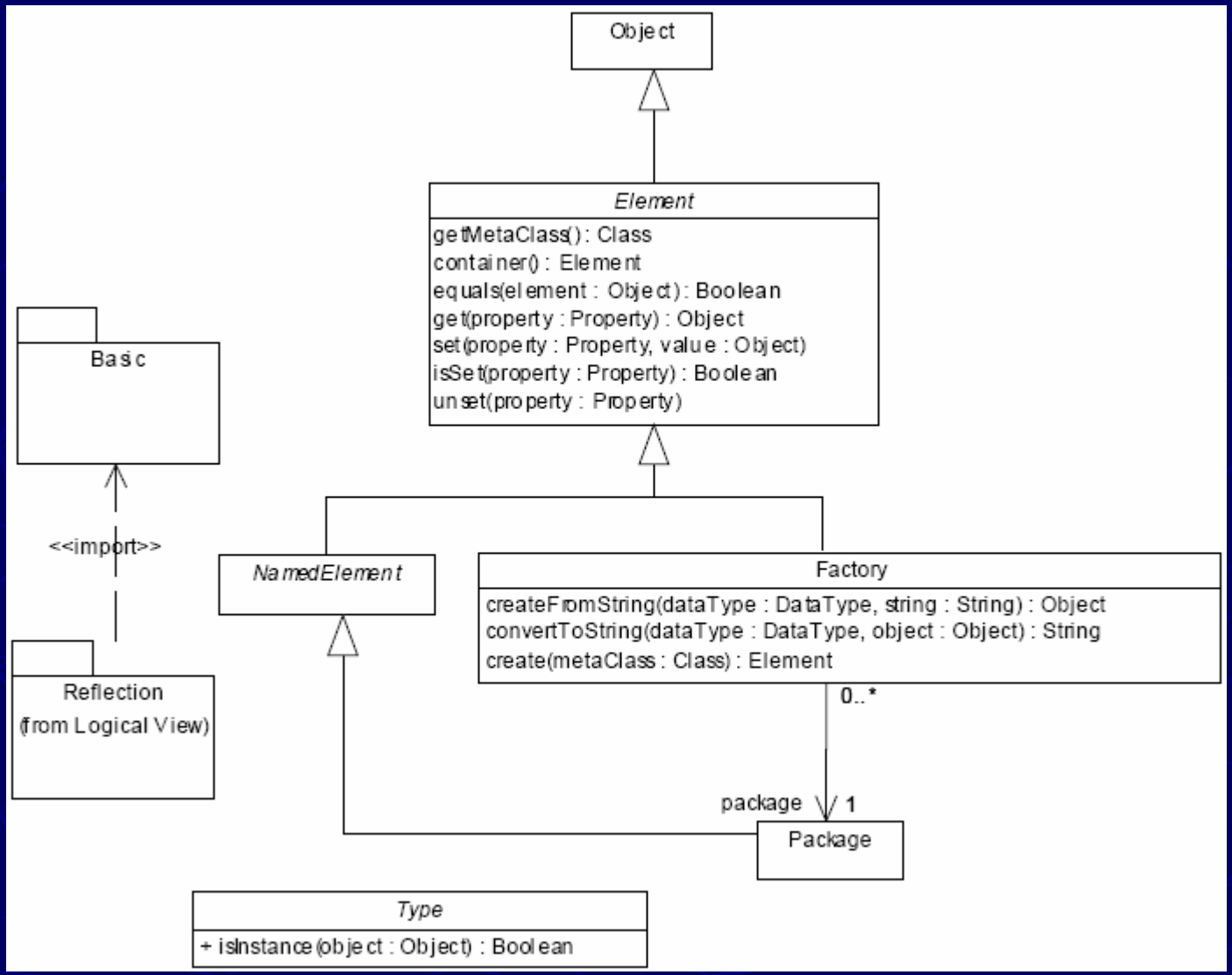
EMOF Types


Element
0..1
0..*
+ownedComment
NamedElement
name : String
+annotatedElement
0..*
Comment
body : String
TypedElement
+type
0..1
Type
isInstance(element : Object) : Boolean

Reflection

- An advantage of metaobjects generally is that they enable use of objects without prior knowledge of the objects' specific features.
- In a MOF context, an object's class (i.e it’s metaobject) reveals the nature of the object
  - kind, and features
- The Reflection Package allows this discovery and manipulation of metaobjects and metadata.

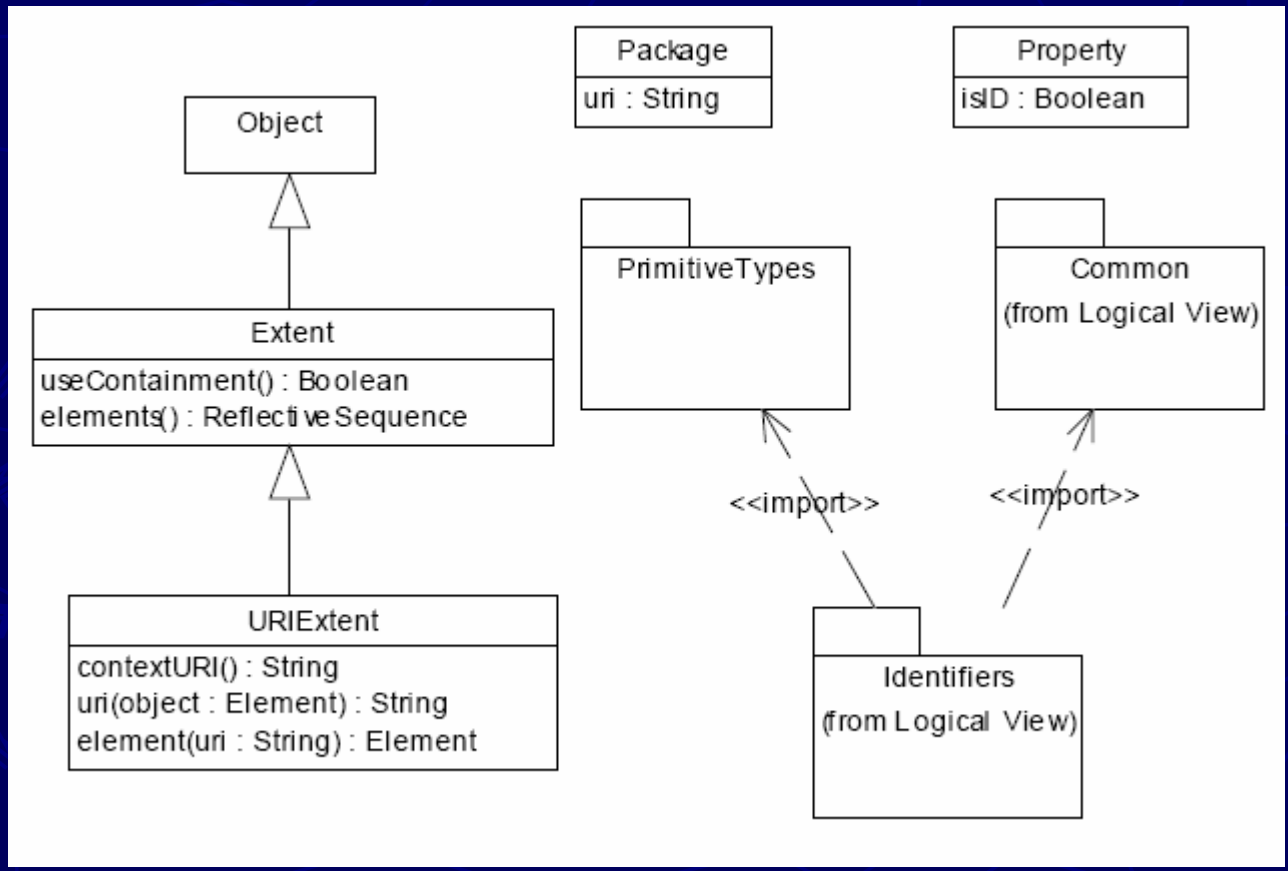
Reflection package



Uniform Resource Identifier

- A URI is the universally unique identification of the package following the IETF URI specification, RFC 2396
  - http://www.ietf.org/rfc/rfc2396.txt

Identifiers Package



Extension

- Simple mechanism to associate a collection of name-value pairs with model elements
- Annotate model elements with additional, perhaps unanticipated, information
  - Information missing from the model
  - Data required by a particular tool

Extension Package