OCL omissions and contradictions

Edward Willink

OCL RTF chair,
QVT RTF representative
Eclipse OCL Project Lead,
Eclipse QVTd Project Lead,

OMG ADTF

21st March 2012

Overview

- Background/Goals for OCL '2.5'
- Values - UML alignment
  - Problems and Solutions
- Types - UML alignment
  - Problems and Solutions
- [Operations - UML alignment
  - Problems and Solutions]
- Summary

OCL 2.5 Goals

- UML 2.5 aligned
  - consistent
- Modeled OCL 'Standard' Library
  - extensible, third party/domain libraries
- Small Core Language + Libraries
  - move Message, State support to library operations
- Fully modeled
  - 100% auto-generated Frame specification (MOFM2T)
  - auto-generated tooling (prototype in new Eclipse OCL)

true = true ?

- OCL 2.3: No Boolean overload of OclAny::=(...)

`OclAny::=(object2 : OclAny) : Boolean`

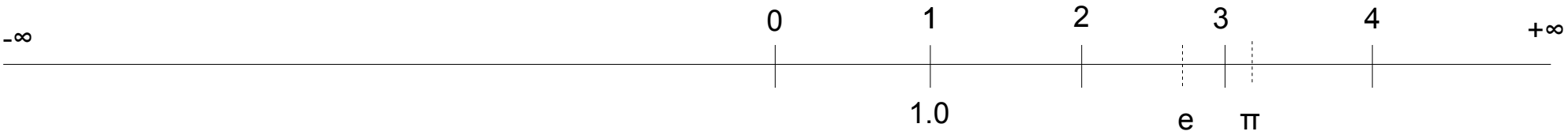
True if self is the same object as object2. Infix operator.

post: result = (self = object2)

- true only equal to true
  - if both trues are the same object
  - no requirement for singleton values
  - no tool does this
- OCL 2.5: Primitive types use value equality

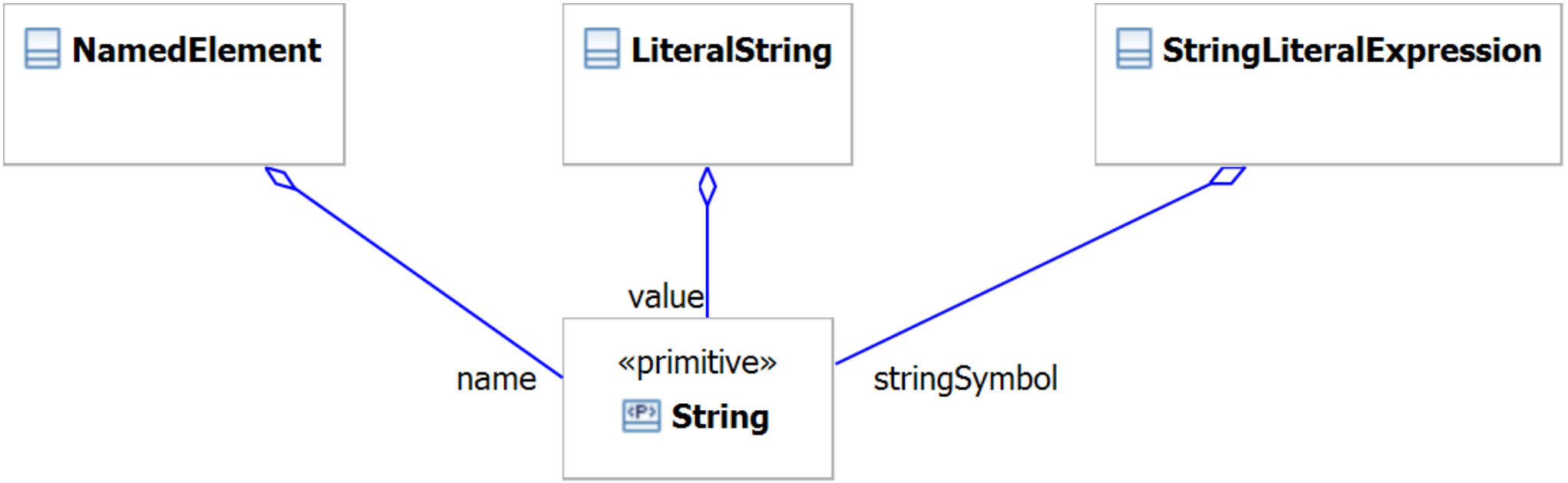
1 = 1.0 ?
Set{1}->including{1.0}->size()
Set{Set{1}}->including{Set{1.0}}->size()

- OCL 2.3: No numeric overload of OclAny::=(...)
- [Java primitives equal, objects not equal]
- OCL is a specification language
  - numbers are points on an infinite number line



- OCL 2.5: Numbers use numeric equality

UML Primitive Usage



- Primitives (Boolean, Integer, Real ...) have no
  - behaviour, representation, conformance
- Representation provided by host Class
- Behaviour/Role defined by host Class

OCL Primitive Usage

- StringValue hosts the String primitive
  - provides a representation
- OCL Standard Library defines behaviour
  - operations of StringValue
    - not String, not a companion class
- OCL specification defines conformance

OCL normalized/pivot values

- Already in OCL 2.0
  - no specified API
- No
  - BooleanValue
  - IntegerValue
  - RealValue
  - OrderedSetValue
- Irregular
  - ...TypeValue

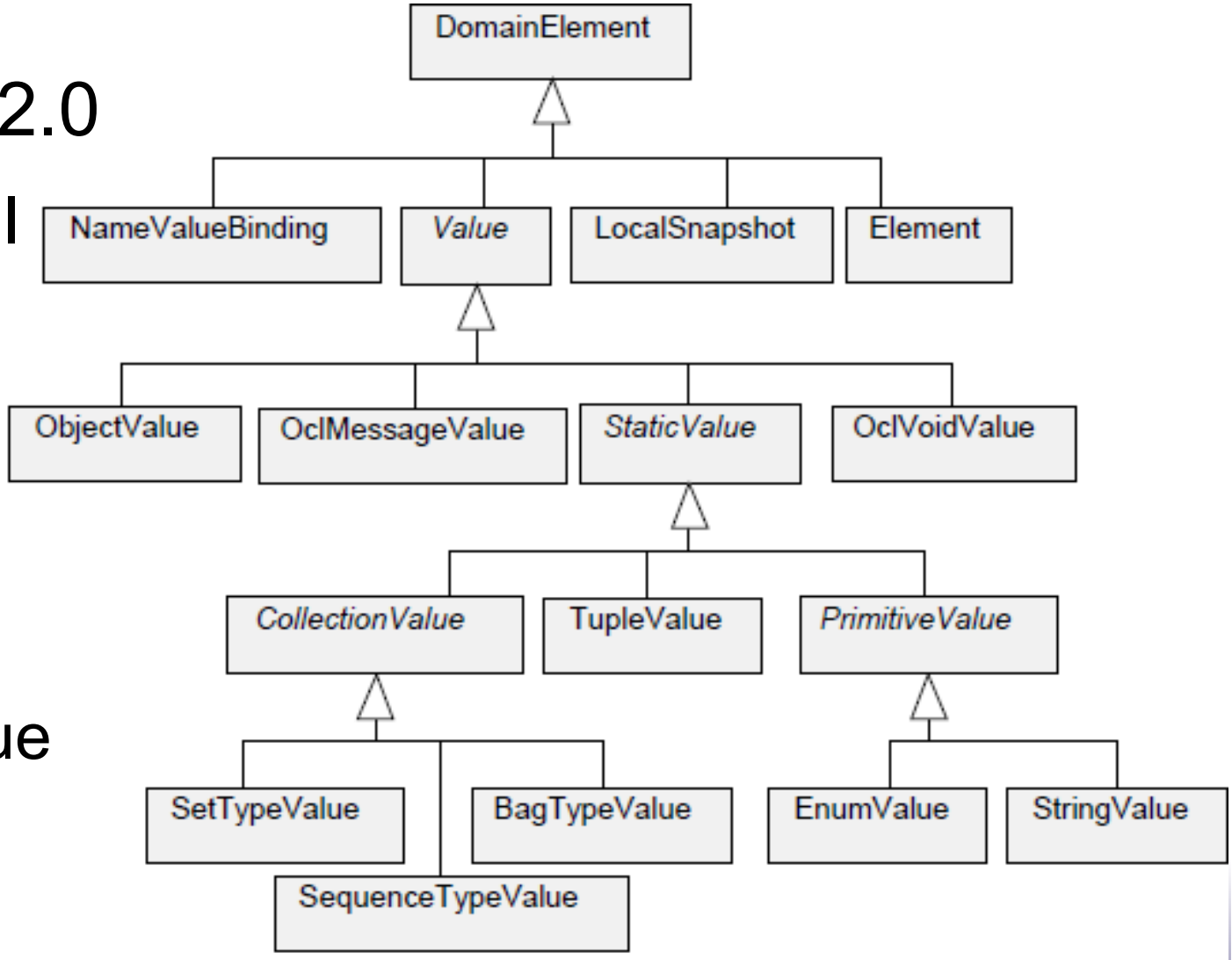
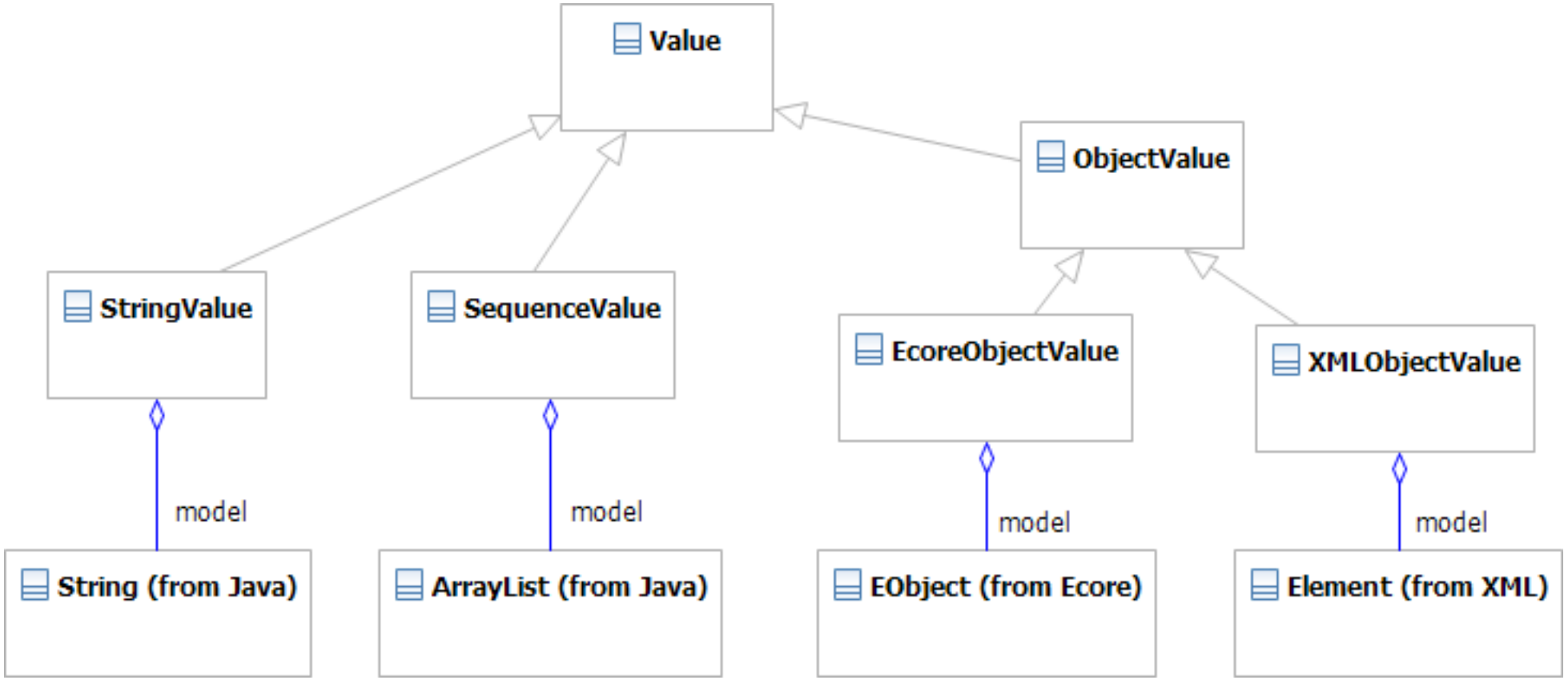


Figure 10.5 - The inheritance tree of classes in the Values package

OCL 2.5 Pivot Values

- Add missing classes
  - BooleanValue, IntegerValue, RealValue, ...
- Regularize names
  - *TypeName*Value
- Promote Values as a run-time API
  - make Value class usage a Compliance Point

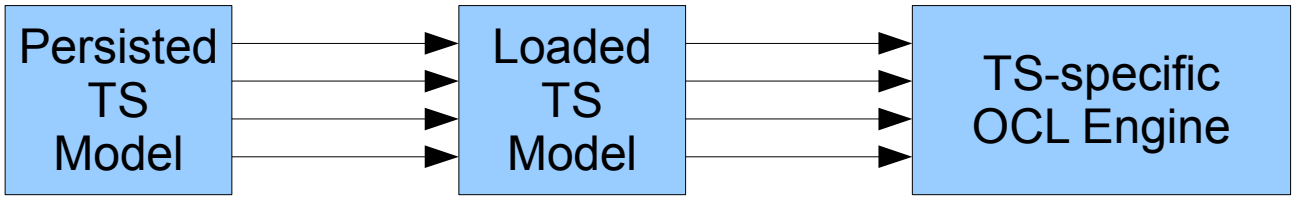
OCL Object Values in Java



- ObjectValue can be polymorphic
  - an indirection to a real Object representation
- Foundation for a Java binding for OCL

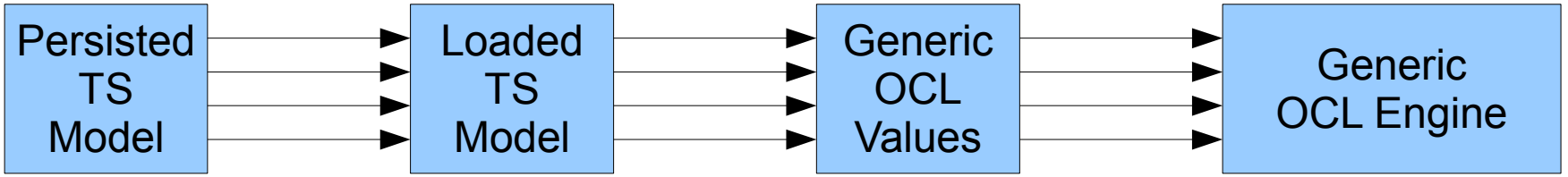
OCL re-use in a Technology Space

- OCL can be used in many Technology Spaces
  - UML, EMOF, XML, RDB, ...
- 'Optimum' re-use
  - use a TS-specific OCL implementation



- Difficult: TS/OCL semantics differences
  - 1 = 1.0, Set{1,1.0} = Set{1}
- Bespoke, irregular, confusing OCL

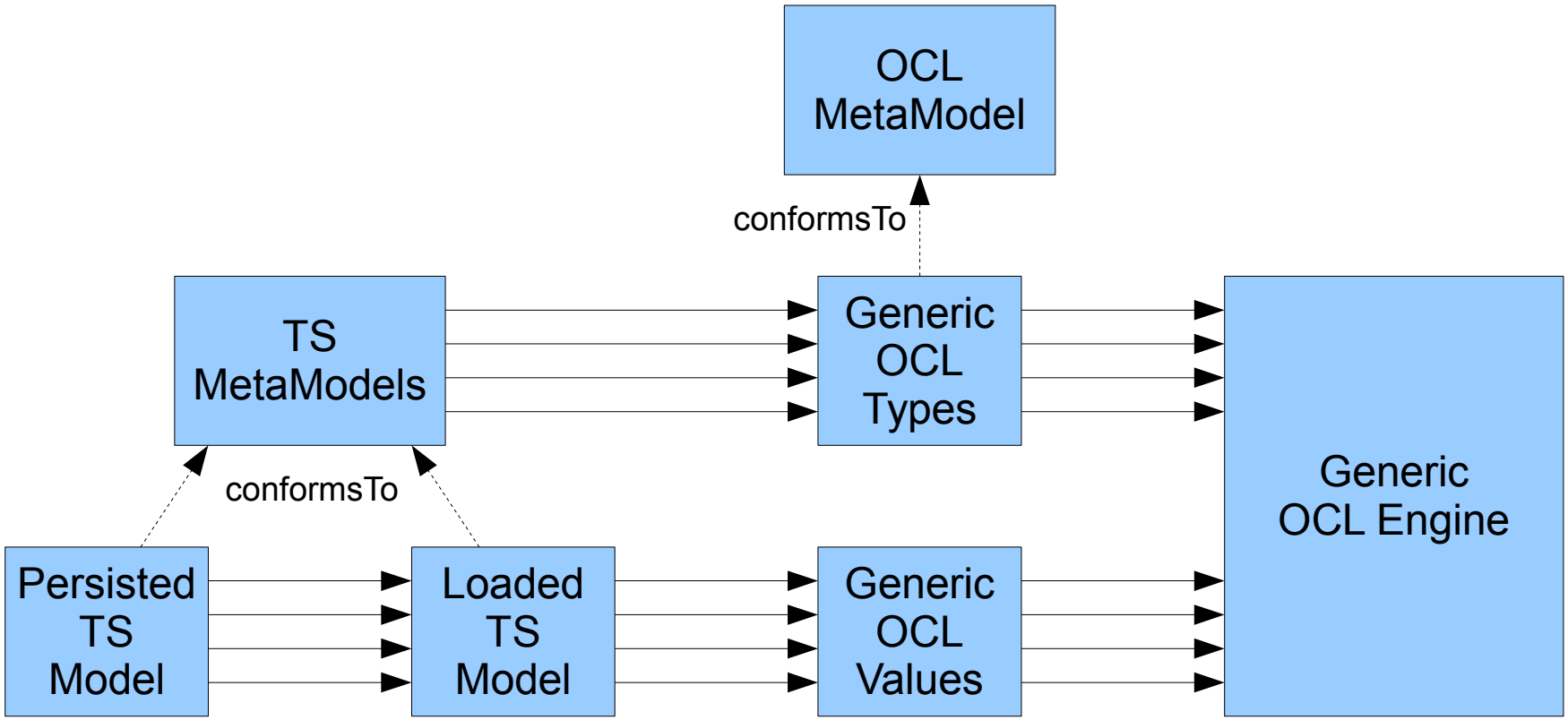
OCL re-use for a Technology Space



- Model accesses convert TS to OCL Values
  - may be faster
    - + no correction for non-OCL TS semantics
    - - a TS to OCL value conversion
  - lazy collection value conversions

- OCL Values should be a conformance point

OCL Types, Meta-model

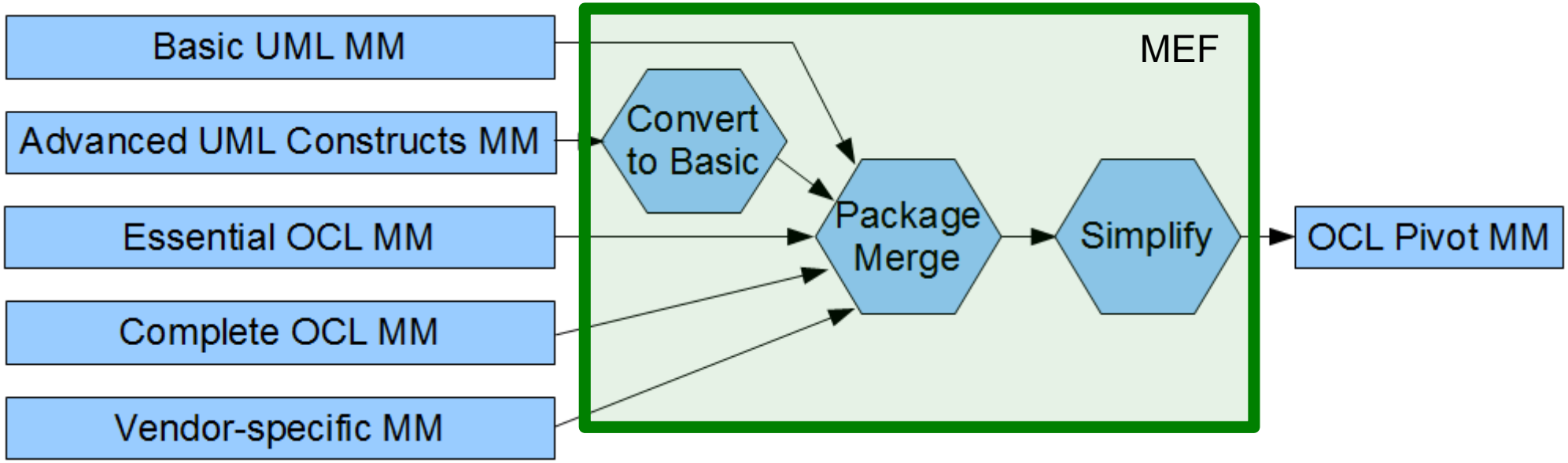


- How is an OCL type aligned to a UML type?
  - not currently specified

OCL Model Elements

- Multiple sources of M2 model elements
  - UML meta-model
  - Essential OCL meta-model
  - Complete OCL meta-model
- Multiple sources of M1 model elements
  - OCL 'Standard' Library
  - User meta-models
  - Complete OCL documents
  - M2 elements by reflection - OclAny::oclType()

M2 Integration



- Selected UML contributions for UML-alignment
- [Selected] OCL contributions [for tailored OCL]
- Vendor contributions support practical tooling
- Merge gives single uniform package
  - OCL::Class, OCL::OclExpression etc

Example OCL M2 'Merges'

- OCL::Class = UML::Class+UML::Classifier+UML::Type
  - in OCL any type can have operations
- Eliminate derived unions
  - simple efficient relationships as in UML Basic
- Full navigability
  - all association ends are Class-owned
- Eliminate non-'MOF' classes
  - no UseCases, Actions, ...
- Re-use UML generalization as OCL conformance

Example OCL M1 'Merges'

- Load user meta-models/libraries
  - from UML/EMOF/... Complete OCL, OCLstdlib
  - normalized to OCL meta-model representation
- Insert OclElement as supertype of all user types
- Co-ordinate package clashes by URI
- Unify duplicate specialisations e.g. Set(String)

OCL Operations

- What are the operation overloading semantics?
  - OCL: aligned to UML
  - UML: implementation variation point
  - therefore unspecified
- Proposal: Java-like invariant overloading
  - A::y(Z) is overloaded by B::y(Z) if B extends A

OclSelf

- OclSelf is the statically determinate type of self
- Self-variant overloading

  Boolean::=(OclSelf) overloads OclAny::=(OclSelf)
  - enhanced invariant overloading: dispatch on
    - dynamic most derived common type of source, argument
  - handles the common binary case
  - Real::+(OclSelf) always gets 2 Real arguments
  - Integer::+(OclSelf) always gets 2 Integers
  - no need for type checks

OCL Iterations

```
aCollection->iterate(q : String | acc : String = '';
   if acc.size() > 0 then acc + ' ' else acc endif + q.toString())
```

- How is `iterate` modeled?
- Iteration is an Operation with extra 'parameter's
  - iterator(s), `q : String`
  - accumulator, `acc : String = ''`
  - body is a lambda-expression `if acc ... + q.toString()`
- Collection types are templates
- Operations/Iterations may be templates

```
type Collection<T> : CollectionType conformsTo OclAny {
 iteration iterate<Tacc>(i : T; acc : Tacc | body : Lambda T() : Tacc) : Tacc
```


Meta-Model Imports

- In Complete OCL
  - import a URI element

```
import uml : 'UML.uml#_jEB8EDoXEeCmpu-HRutBsg'
```

  - import a URI element and its child names

```
import uml : 'UML.uml#_jEB8EDoXEeCmpu-HRutBsg'.*
```

- In Essential OCL
  - no mechanism to contextualize expression
  - use URI as a path name element

```
oclIsKindOf('UML.uml#Activities'::Action)
```


Unified Collection Types

- UML: bounded collection types
  - String[2..4] {ordered}
  - no nested collection types
- OCL: nested collection types
  - Set(OrderedSet(String))
  - no bounded collection types
- Inconsistent primary/secondary declarations
- Unified Collection Types
  - Set(OrderedSet(String)[2..4])

Reflection

OCL 2.0 **`Element::getMetaClass()`**

- MOF facility not merged to UML

OCL 2.2 **`OclAny::oclType() : Classifier`**

- No Classifier in EMOF
- Classifier at different meta-level

- Does OCL support Reflection?
  - (OCL 2.0) precondition for **`Sequence::first()`**

```
self.oclType().elementType.oclIsKindOf(CollectionType)
```

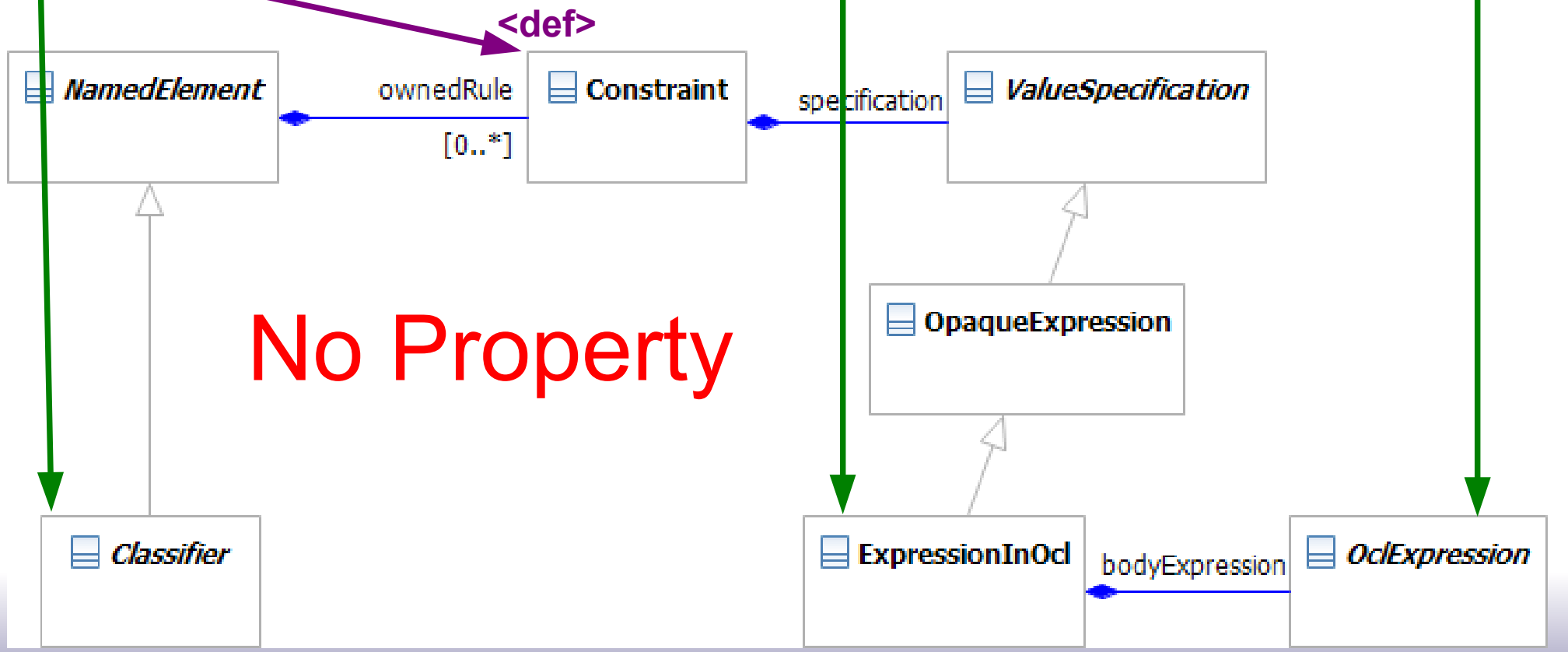
- Yes

Unlimited / Plus infinity

- [0..*] multiplicity upper bound can be unlimited
  - * is not an Integer
  - new UnlimitedNatural type (and value)
  - UnlimitedNatural is not a simple subtype of Integer
    - *.oclAsType(Integer) is invalid
    - requires run-time test
- OCL 2.5: * is plus infinity
  - Integer and Real support +/- infinity
  - all UnlimitedNatural values are Integer values

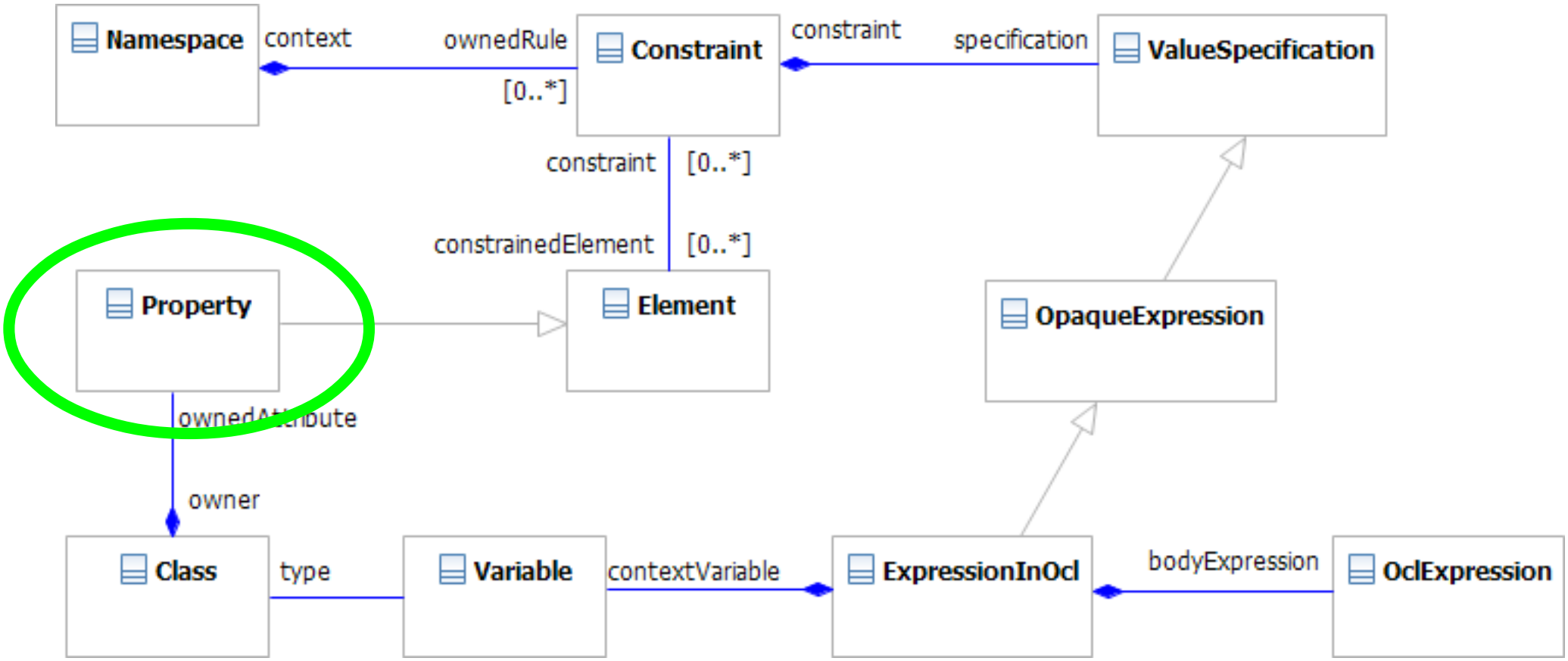
Complete OCL 2.3 Property Definition

```
context MyClass
def: upperCaseName : String = name.toUpper()
```



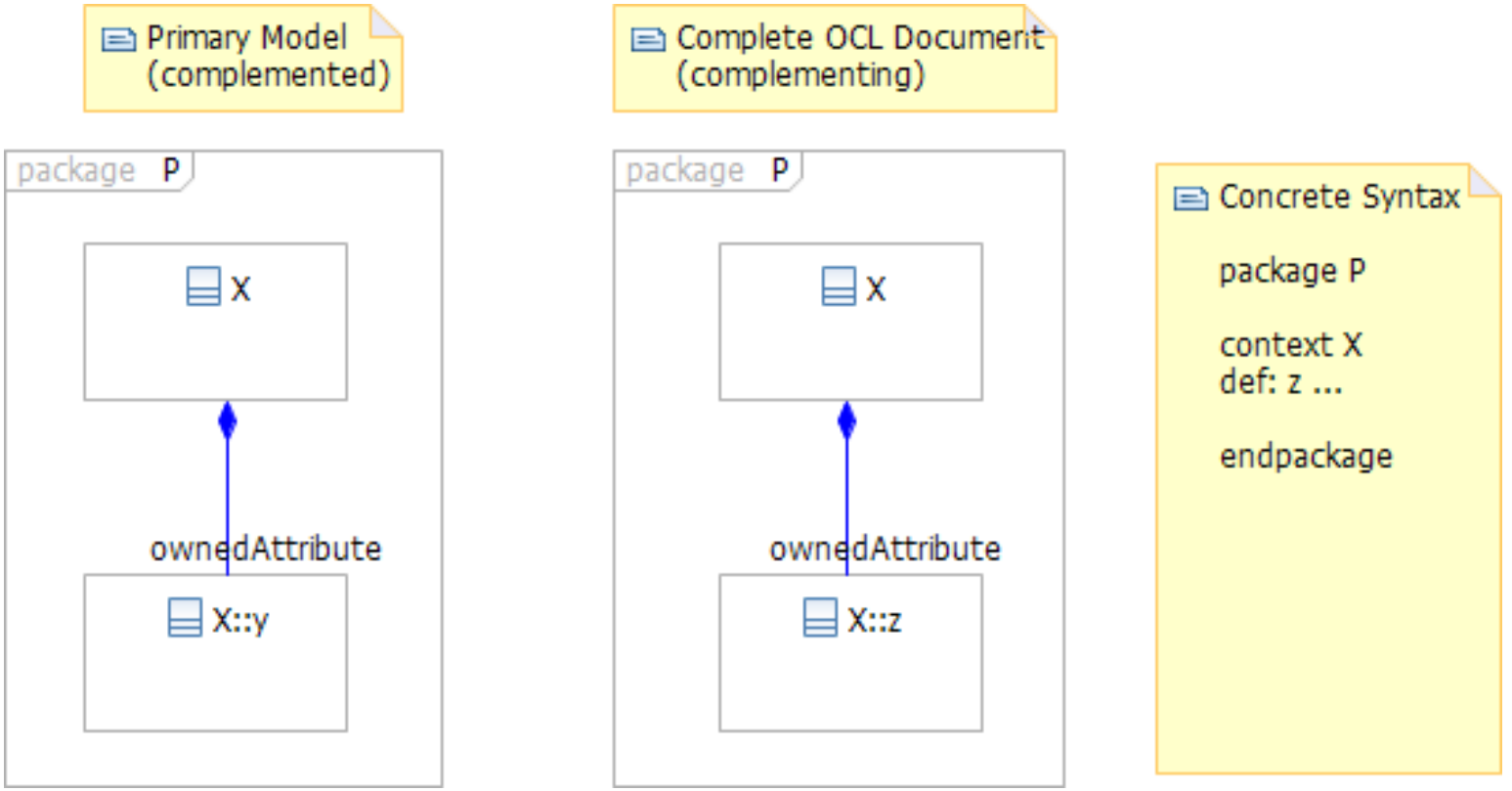
PropertyCallExpression.referredProperty impossible in XMI

New Complete OCL Property Definition



- Property defined (can be referenced)

New Multiple Models Problem



- P::X::y defined by a primary UML model
- P::X::z defined by a Complete OCL document

- How many P's? How Many X's?
- What is the value of P::X.ownedAttribute?

Meta-Model Problems to solve

- Library Modeling
  - Reflection
  - Iteration
- UML alignment
  - obsolete/inconsistent classes in use
  - templates
- EMOF (and Ecore and ...) utility
  - arbitrary meta-meta-models
- Complete OCL realizability
  - definition of real Property/Operation
  - multiple models, URIs

Multiple Models Solution

- Simple Model Usage
  - How many P's? How Many X's?
    - One
  - What is the value of P::X.ownedAttribute?
    - Set{P::X::y, P::X::z}
- Reflective Model Usage, URI Access
  - How many P's? How Many X's?
    - Two
  - What is the value of P::X.ownedAttribute?
    - depends on P::X - Set{P::X::y} or Set{P::X::z}

Summary- Not-new OCL Facilities

- Overloading / dynamic dispatch
- Reflection : oclType(), Class<T>
  - was T, used in WFRs
- Type-valued Expressions : Class<T>
  - used in oclAsType()
- Templates/Generics
  - used in Collection, Collection::product, Tuples
- OclSelf
  - was T
- Lambda Types / Expressions
  - iteration bodies

UML Primitives

- UML Primitives have no representation
  - ensures implementation freedom
- UML Primitives have no behaviour
  - cannot be used
- UML Primitives have no conformance
  - cannot be interchanged

OCL re-use

- How does OCL support UML, EMOF, XML, ... ?
  - Customize OCL for each technology space
    - bespoke, irregular, confusing OCL
  - Map each technology space to OCL
    - most of OCL is technology space neutral
    - just model access to align - values and types
    - partial mapping gives partial support
- e.g.EMOF has no Associations
    - EMOF to OCL mapping may reconstruct associations
    - an EMOF problem, not an OCL problem

Overview

- OCL and UML and EMOF
  - Problems
- Complete OCL
  - Problems and Solutions
- Values
  - Problems and Solutions
- Summary

OCL Values in Java

- xxxValue provides an indirection

- IntegerValue/SetValue provide OCL semantics
  - exploit Java for implementation not behaviour
- IntegerValue may be polymorphic
  - int or long or BigInteger representation
  - IntegerValue-for-int similar to java.lang.Integer

LocalSnapShots

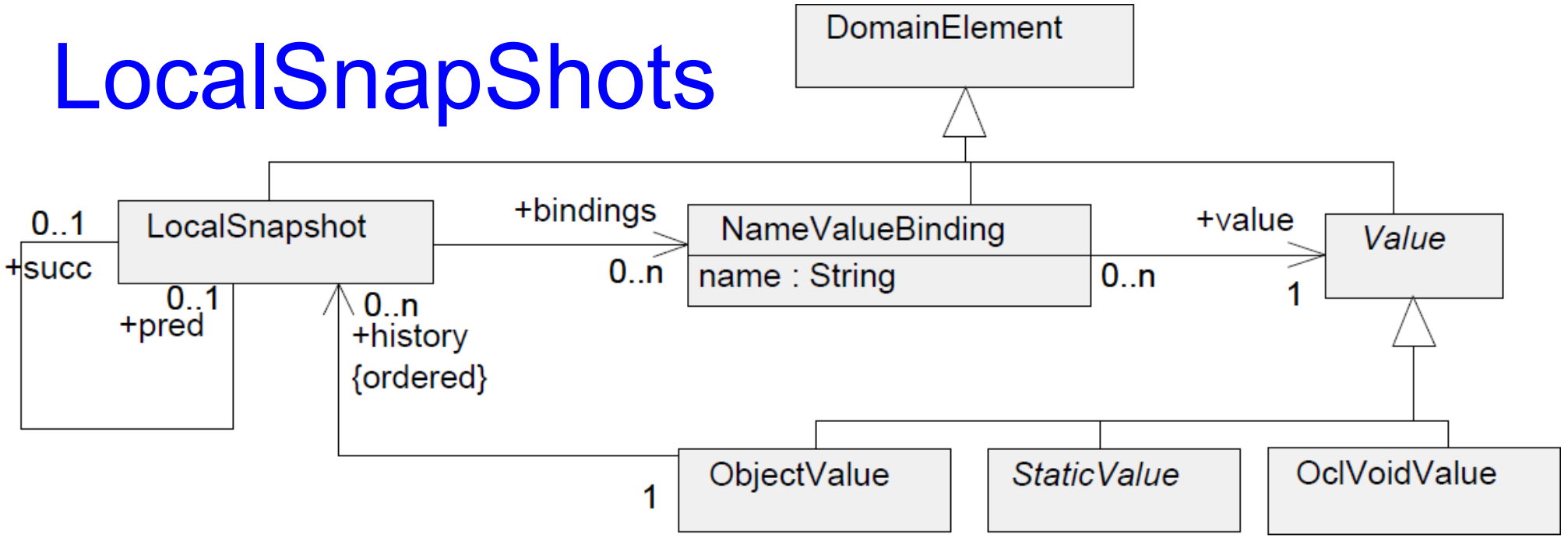
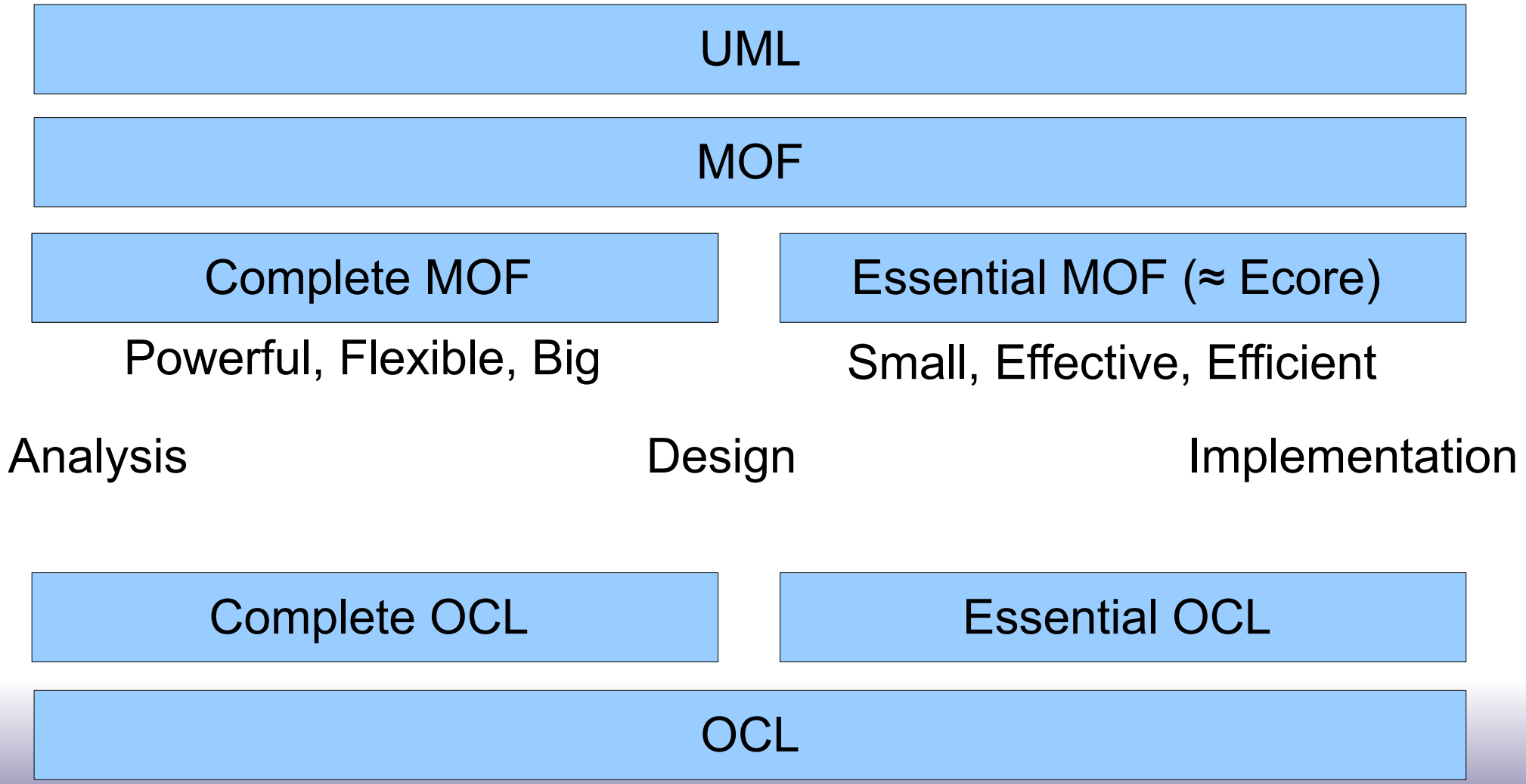


Figure 10.2 - The kernel values in the semantic domain

- LocalSnapShot: all names and their values
  - useful for defining semantics
    - N snapshots for OclMessage history
    - 2 snapshots necessary for @pre, not @pre
  - inefficient for practical implementations

UML (and OCL)



Significant Problem Summary

- OCL for EMOF lacks essential elements
  - OpaqueExpression, Constraint
  - Types with features
- Complete OCL incomplete
  - Property/Operation definition not useable
  - Property/Operation definition not persistable
- OCL not UML aligned
  - AssociationEnd/Property
  - AssociationClass

Solution Summary

- OCL for EMOF lacks essential elements
  - support EMOF only indirectly
- Complete OCL incomplete
  - complete it
- OCL not UML aligned
  - define OCL with respect to UML

Values

- Integer ??
- Type ??
- OrderedSet ??
- *Type* ??

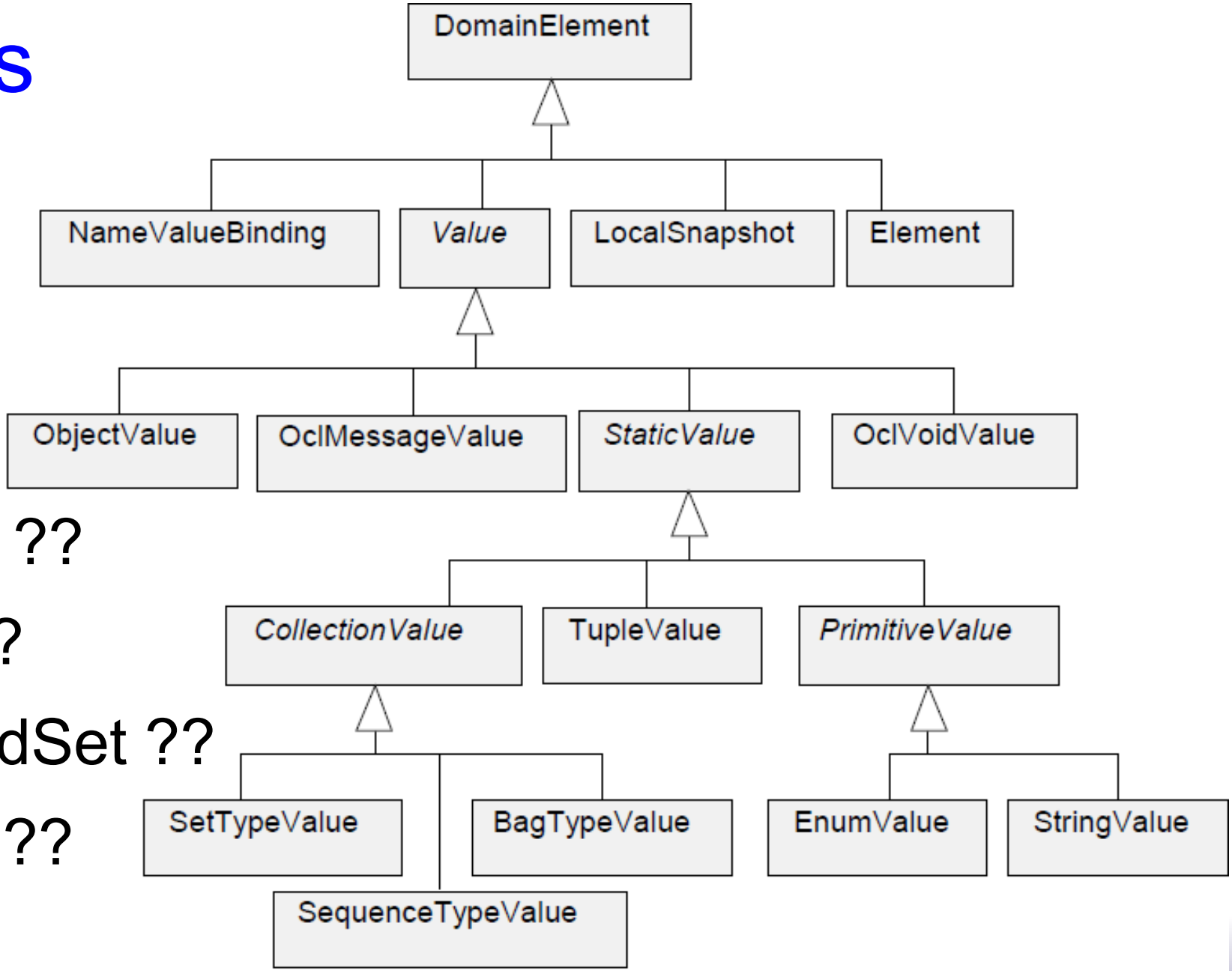
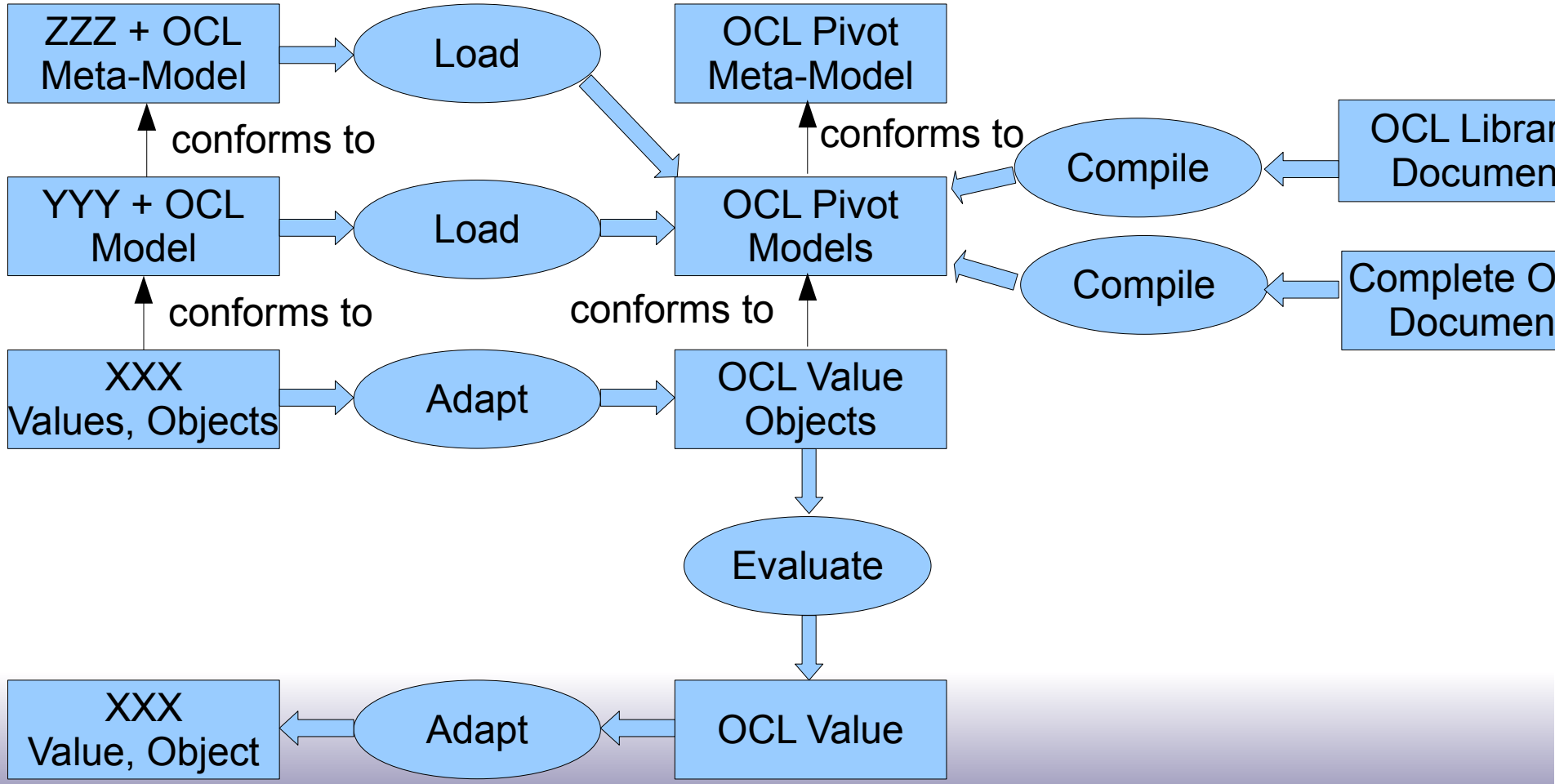


Figure 10.5 - The inheritance tree of classes in the Values package



UML and OCL integration