OCL Advances and the OCL VM

Edward Willink

OMG OCL RTF
Eclipse OCL Project Lead

OMG QVT RTF
Eclipse QVT Declarative Project Lead

Eclipse/OMG Workshop
Reston
25th March 2012

Overview

- Why OCL?
- Recent Advances
  - Embedded OCL with the OCLinEcore editor
  - Independent OCL with the Complete OCL editor
- OCL Virtual Machine
  - Flexible Accurate Values
  - Fast Operations
  - Fast Scheduling
- Summary

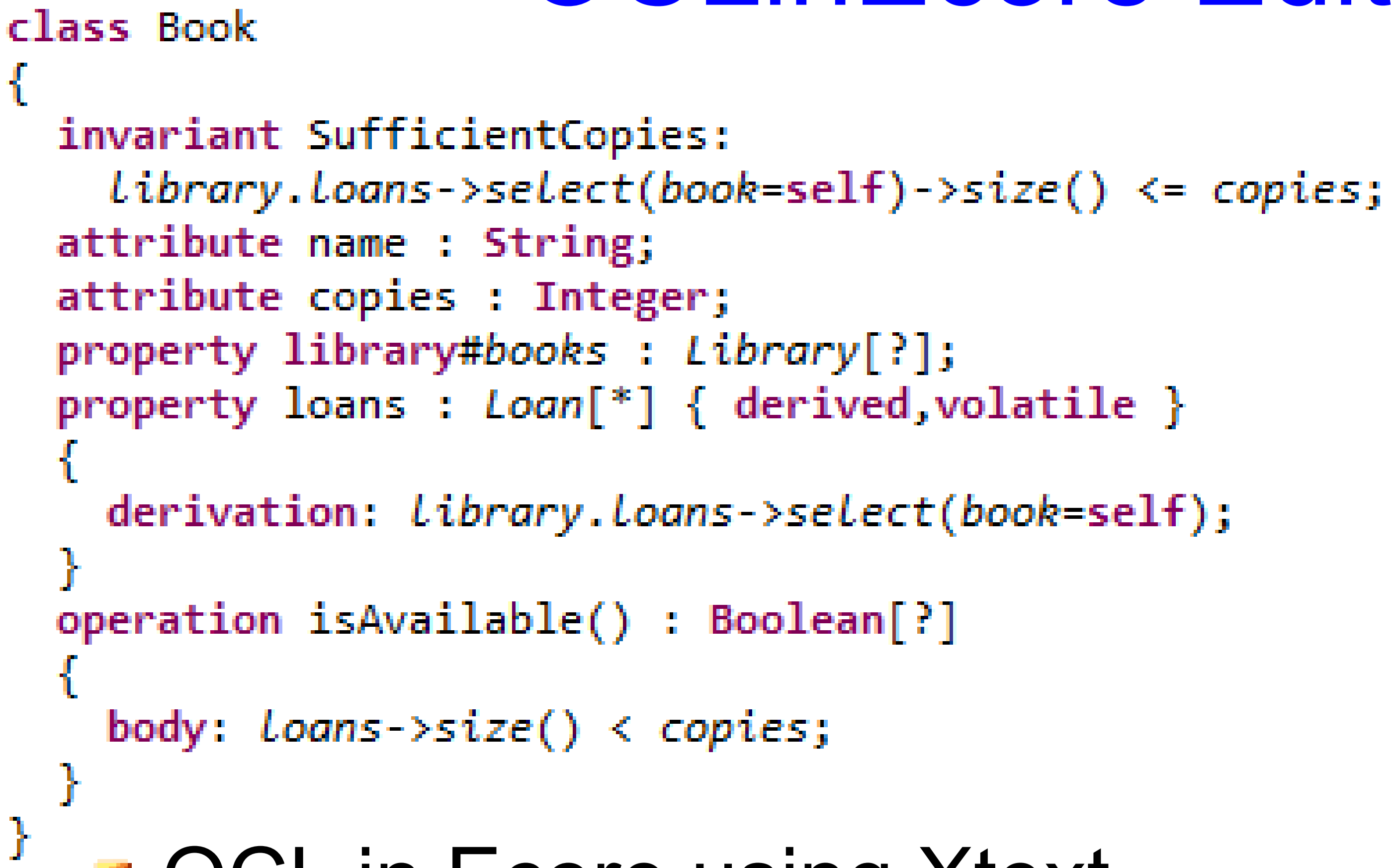
OCL or Xbase?

- Xbase
  - very good, well supported
  - tied to Java as an implementation platform
- OCL
  - specification language
  - so what?

Java gets it wrong

- 1 not always equal to 1.0 in Java
  - Set{1, 1.0} may have two elements
- Must use BigInteger for unlimited numbers
- Has assignment
  - uncontrolled side effects
  - not declarative
  - cannot be analyzed
  - cannot support optimized re-evaluation
    - Eclipse OCL introduced an Impact Analyzer (Indigo)

OCLinEcore Editor



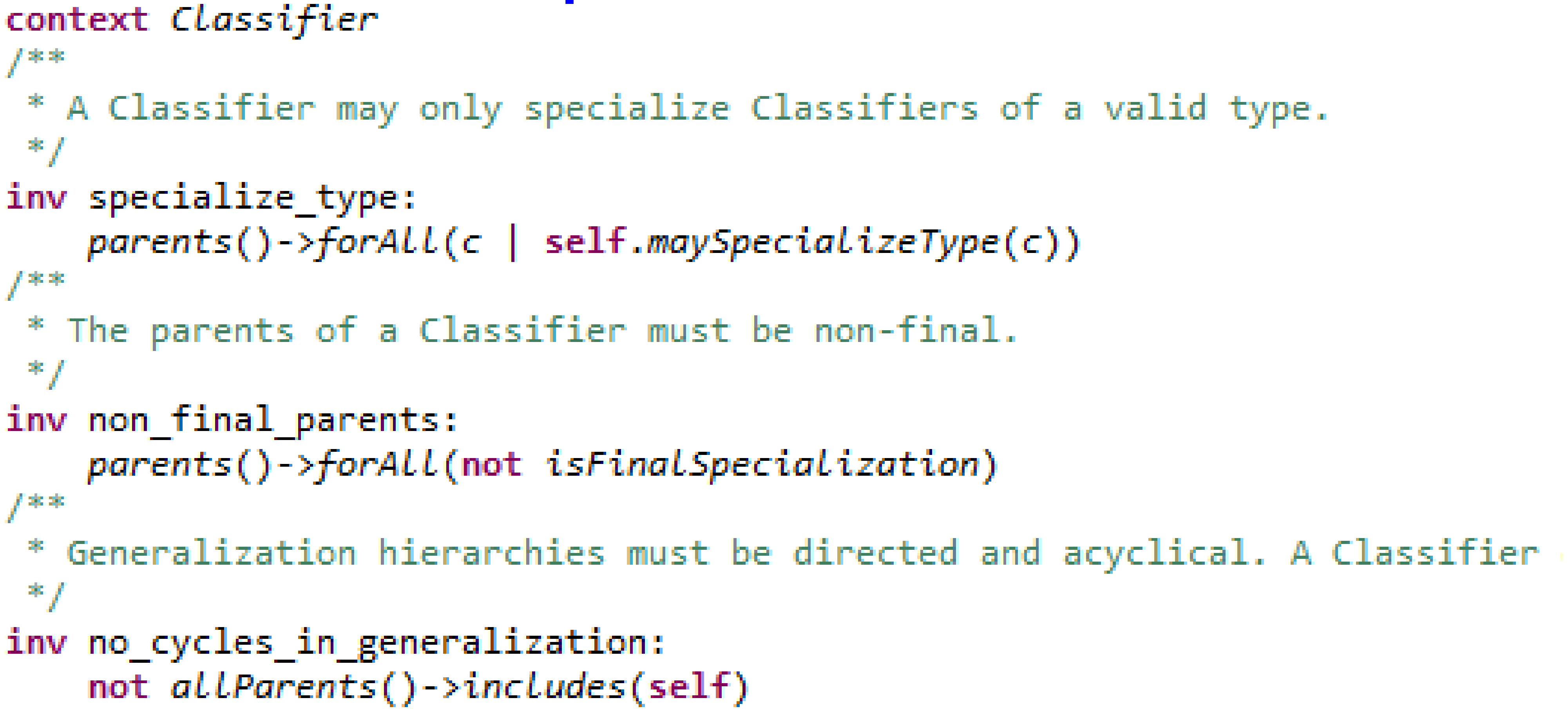
```
class Book
{
  invariant SufficientCopies:
    library.loans->select(book=self)->size() <= copies;
  attribute name : String;
  attribute copies : Integer;
  property library#books : Library[?];
  property loans : Loan[*] { derived,volatile }
  {
    derivation: library.loans->select(book=self);
  }
  operation isAvailable() : Boolean[?]
  {
    body: loans->size() < copies;
  }
}
```

- OCL in Ecore using Xtext
  - persist directly as *.ecore, or as *.oclinecore
- Checked, readable, accessible constraints

OCL in Ecore usage

- Technology: EAnnotations with delegate URIs
- Invariants (EAnnotation for EClass)
  - executed as part of EMF Validation
    - e.g. Validate action in any Ecore editor
- Derived/Initial Properties
  - executed as part of eGet('XX'), getXX
- Operation bodies
  - executed as part of eInvoke('YY'), yy()
- Provided OCL plugins are installed

CompleteOCL Editor

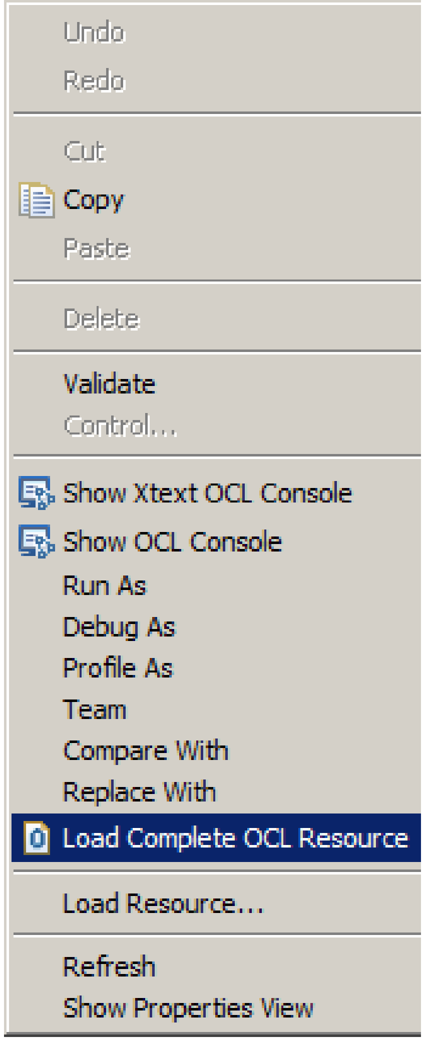


```
context Classifier
/**
 * A Classifier may only specialize Classifiers of a valid type.
 */
inv specialize_type:
    parents()->forAll(c | self.maySpecializeType(c))
/**
 * The parents of a Classifier must be non-final.
 */
inv non_final_parents:
    parents()->forAll(not isFinalSpecialization)
/**
 * Generalization hierarchies must be directed and acyclical. A Classifier
 */
inv no_cycles_in_generalization:
    not allParents()->includes(self)
```

- Complementary OCL document
  - persist as *.ocl

Complete OCL Usage

- Complementary independent document
  - not known to complemented model/tooling
  - pre-Juno: requires manual Java loading
- Juno: "Load Complete OCL Resource"
  - wherever a ResourceSet is accessible
    - Sample Ecore Editor/EMF Generated Editor
    - Xtext Editor/Xtext Generated Editor



  - impose style checking - uppercase terminals
  - diagnose bad usage - all references have opposites

Specification Tooling

- OCL tooling now useable
  - OCLinEcore for primary models with OCL
  - Complete OCL for secondary OCL
- OCL 2.5 specification auto-generated from
  - Xtext annotated EBNF grammars
  - UML/Ecore + OCL models
- Eclipse OCL 2.5 tooling auto-generated from
  - the same specification models
- Same approach planned for QVT

OCL2Java Code Generation

- Helios, Indigo
  - OCL in Ecore as EAnnotations
    - genmodel: Strings containing unchecked OCL
    - run-time: compile and interpret
- Juno (optional)
  - OCL in Ecore as EAnnotations
    - genmodel: OCL converted to Java code
    - genmodel: dispatch tables for fast execution
    - run-time: direct Java execution by OCL VM

OCL VM: Polymorphic Value Hierarchy

- Everything is-a Value
- Primitive values
  - {Boolean, Integer, Real, String, UnlimitedNatural}Value
- Templated Collection values
  - {Bag, OrderedSet, Sequence, Set}Value
- Infrastructure values
  - {Invalid, Lambda, Null, Tuple}Value
- Model Element values
  - {Object, Type}Value ...

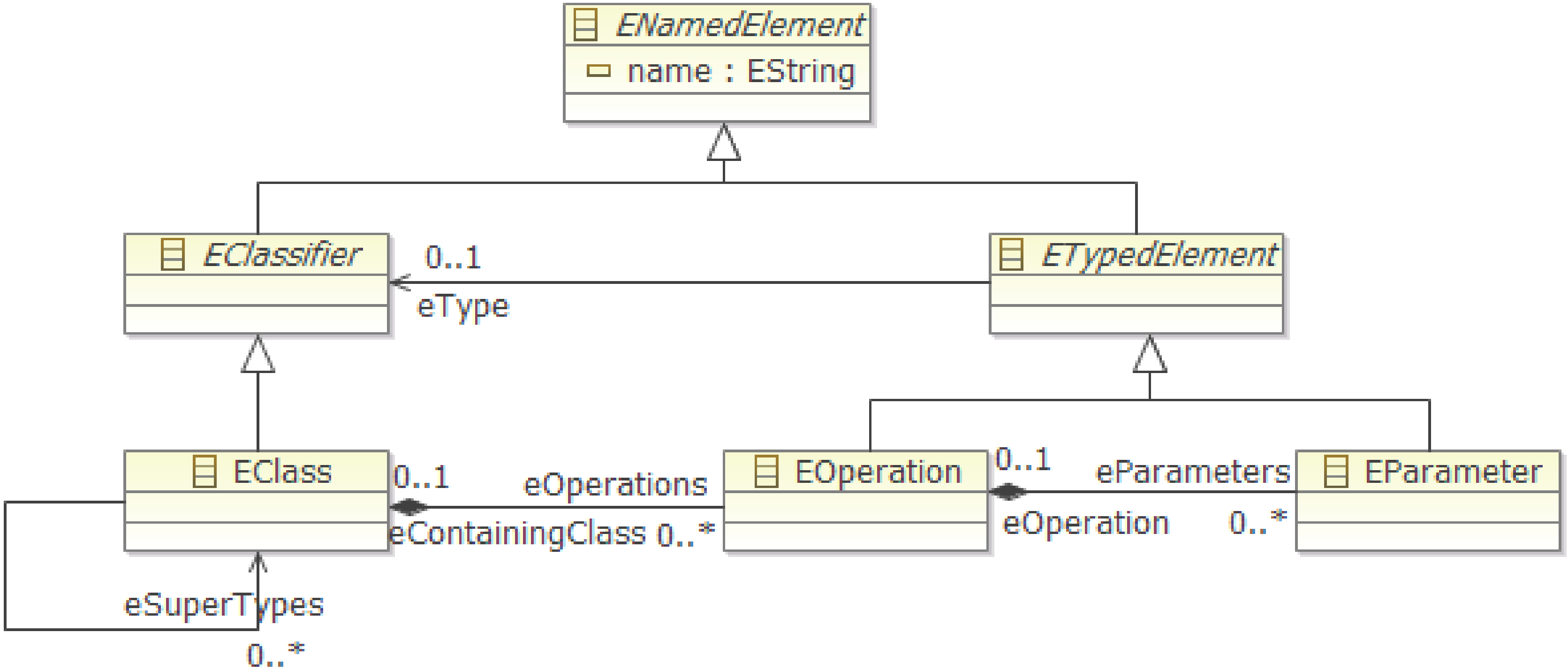
Polymorphic Values

- Integer/Real have unlimited size
  - BigInteger/BigDecimal in Java - not polymorphic
- Multiple implementations of IntegerValue
  - simplest implementation like java.lang.Integer
    - wrapper for int
    - overflow detector for add/subtract/.... => growth
  - bigger implementation uses java.math.BigInteger
  - efficiency of small representation
  - automatic conversion to larger representation

Polymorphic Object Values

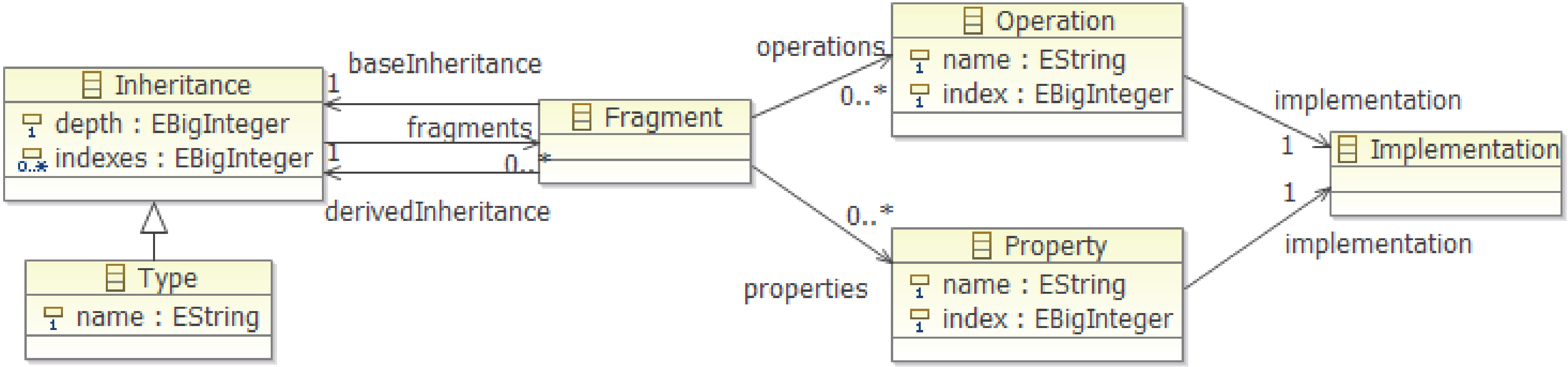
- Objects/Model Elements are also Values
  - but EObject is not a Value
  - so EObjectValue adapts EObject to be a Value
  - 'simple' adapters for other technology spaces
- All model elements normalised to ObjectValue
- Meta-model elements are normalised as well
  - all types are TypeValues

'Ecore' Operation Call : a.b(c,d)



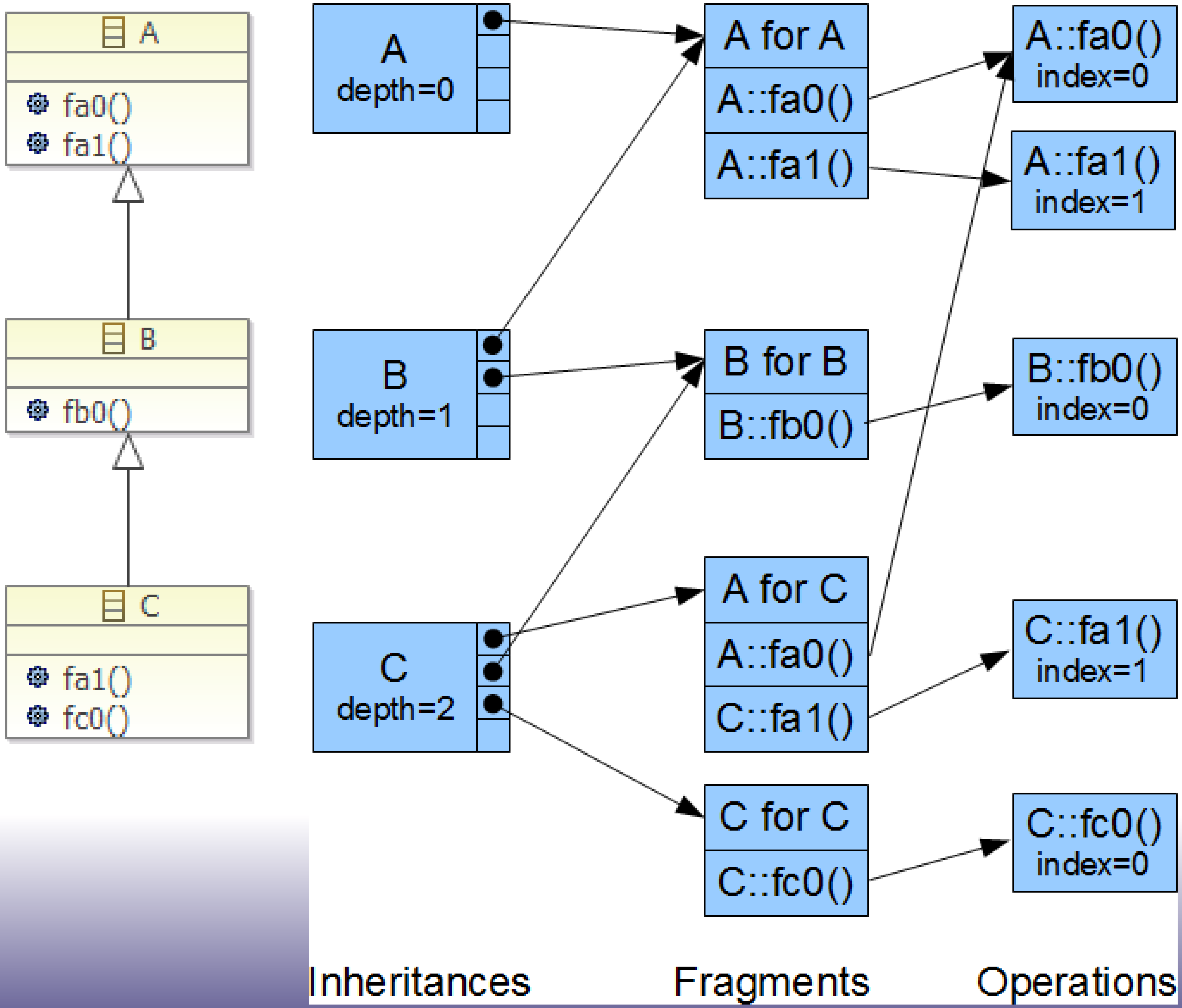
- Tree search over type and supertypes --- a
  - Linear search for operation name --- b
    - Linear search to match argument types --- (c,d)
      - Tree search for conformant type/supertype --- c then d
- Select best unique match

'OCL VM' Operation Call



- Fragment provides derived view of base
  - may have overloaded entries
- Linear search of fragments at required depth
  - Direct index to operation

Example OCL VM dispatch



- Problem
  - fa1() for a C
- Compile-time
  - A::fa1
  - A for ? index 1
- Run-time
  - C
  - A is depth 0
    - A for C
  - A::fa1 is index 1
    - C::fa1

Dispatch comparison

- Direct Ecore
  - potentially 6D search
    - all super classes
    - all operations
    - all parameters
    - all super classes again
- OCL VM Dispatch tables
  - 1D search over width of inheritance tree
    - usually 1, sometimes 2 or 3 steps

Auto-generated to ...Tables.java

```
private static final ExecutorOperation[] _Integer__Real = {
    OCLstdlibTables.Operations._Integer___mul_ /* _'*'(OclSelf) */
    OCLstdlibTables.Operations._Integer___add_ /* _'+'(OclSelf) */
    OCLstdlibTables.Operations._Integer__0__sub_ /* _'-'() */,
    OCLstdlibTables.Operations._Integer__1__sub_ /* _'-'(OclSelf) *
    OCLstdlibTables.Operations._Integer___div_ /* _'/'(OclSelf) */
    OCLstdlibTables.Operations._Real___lt_ /* _'<'(OclSelf) */,
    OCLstdlibTables.Operations._Real___lt__eq_ /* _'<='(OclSelf) */
    OCLstdlibTables.Operations._Real___lt__gt_ /* _'<>'(OclSelf) */
    OCLstdlibTables.Operations._Real___eq_ /* _'='(OclSelf) */,
    OCLstdlibTables.Operations._Real___gt_ /* _'>'(OclSelf) */,
    OCLstdlibTables.Operations._Real___gt__eq_ /* _'>='(OclSelf) */
    OCLstdlibTables.Operations._Integer__abs /* abs() */,
    OCLstdlibTables.Operations._Integer__compareTo /* compareTo(Ocl
    OCLstdlibTables.Operations._Real__floor /* floor() */,
    OCLstdlibTables.Operations._Integer__max /* max(OclSelf) */,
    OCLstdlibTables.Operations._Integer__min /* min(OclSelf) */,
    OCLstdlibTables.Operations._Real__round /* round() */,
    OCLstdlibTables.Operations._Integer__toString /* toString() */
};
```

```
private static final ExecutorFragment[] _Integer =
{
    Fragments._Integer__OclAny /* 0 */,
    Fragments._Integer__OclComparable /* 1 */,
    Fragments._Integer__OclSummable /* 1 */,
    Fragments._Integer__Real /* 2 */,
    Fragments._Integer__Integer /* 3 */
};
private static final int[] __Integer = { 1,2,1,1 };
```

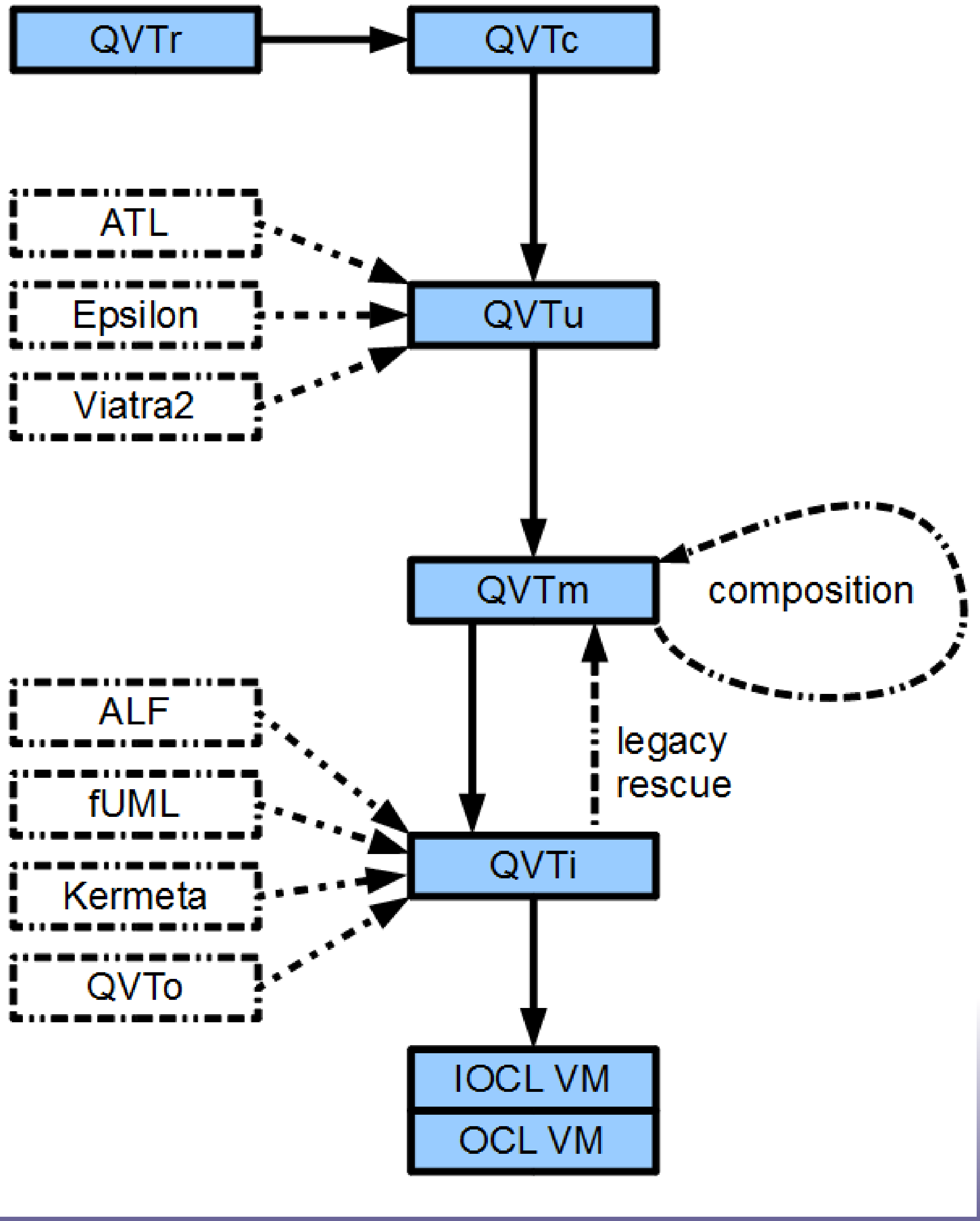
```
static final ExecutorProperty[] _NamedElement = {
   PivotTables.Properties._Element__Comment,
   PivotTables.Properties._Element__Constraint,
   PivotTables.Properties._NamedElement__isStatic,
   PivotTables.Properties._NamedElement__name,
   PivotTables.Properties._NamedElement__ownedAnnotation,
   PivotTables.Properties._Element__ownedComment,
   PivotTables.Properties._NamedElement__ownedRule
};
```

genmodel integration

- OCL Examples & Editors Genmodel Adapter
  - .../XXXTables.java
  - .../bodies/*Body.java
  - .../impl/*Impl.java
  - MANIFEST.MF needs manual edit
    - depends on org.eclipse.ocl.examples.library
- Provided
  - Global OCL Preference
    - "Realisation of OCL embedded in Ecore models" set to "Generate Java code in xxxBodies classes"
  - No "http://www.eclipse.org/OCL/GenModel" GenAnnotation
    - with"Use Delegates" set true

(Imperative) OCL VM for QVT

- Simple OCL VM
  - AST walker
  - QVT richer AST
- Code generated VM
  - dispatch tables
  - flattened code
  - inlined operations
- Debugging tools



Simple Interpreted OCL VM

- Program is an Abstract Syntax Graph (AST)
  - VariableExp
    - references variable to read as a value
  - PropertyCallExp
    - references object and property to read as a value
  - OperationCallExp
    - references operation to apply to some values
- Run-time Interpretation
  - tree-walking evaluation visitor
- Extensible with new AST node classes

Code Generated OCL VM

- Program is an Abstract Syntax Graph (AST)
- Compile-Time Code Generation
  - tree-walking code generating visitor
- Run-time Execution
  - direct Java, direct model accesses
- Extensible with new AST node classes
- Optimisable
  - direct model access getXX() rather than eGet('XX')
  - inlining of non-polymorphic (final) operations

Summary

- OCL Tooling
  - editors ready for specification usage
- OCL VM
  - polymorphic specification value system
  - normalising adapters to practical objects
  - efficient tables for normalised meta-models
  - fast polymorphic dispatch of operations
  - foundation for QVT and other Tx languages