

# Enriching Your Models with OCL

Adolfo Sánchez-Barbudo Herrera, Open Canarias

Axel Uhl, SAP

Edward Willink, MDT/OCL Project Lead



3<sup>rd</sup> November 2010



canarias  
OBJETIVO de PROGRESO



Unión Europea  
Fondo Social Europeo



# Overview

## MDT/OCL team

- Why and When OCL
- Introduction to OCL
- OCL within Eclipse
- OCL Use Cases, coming soon

## SAP

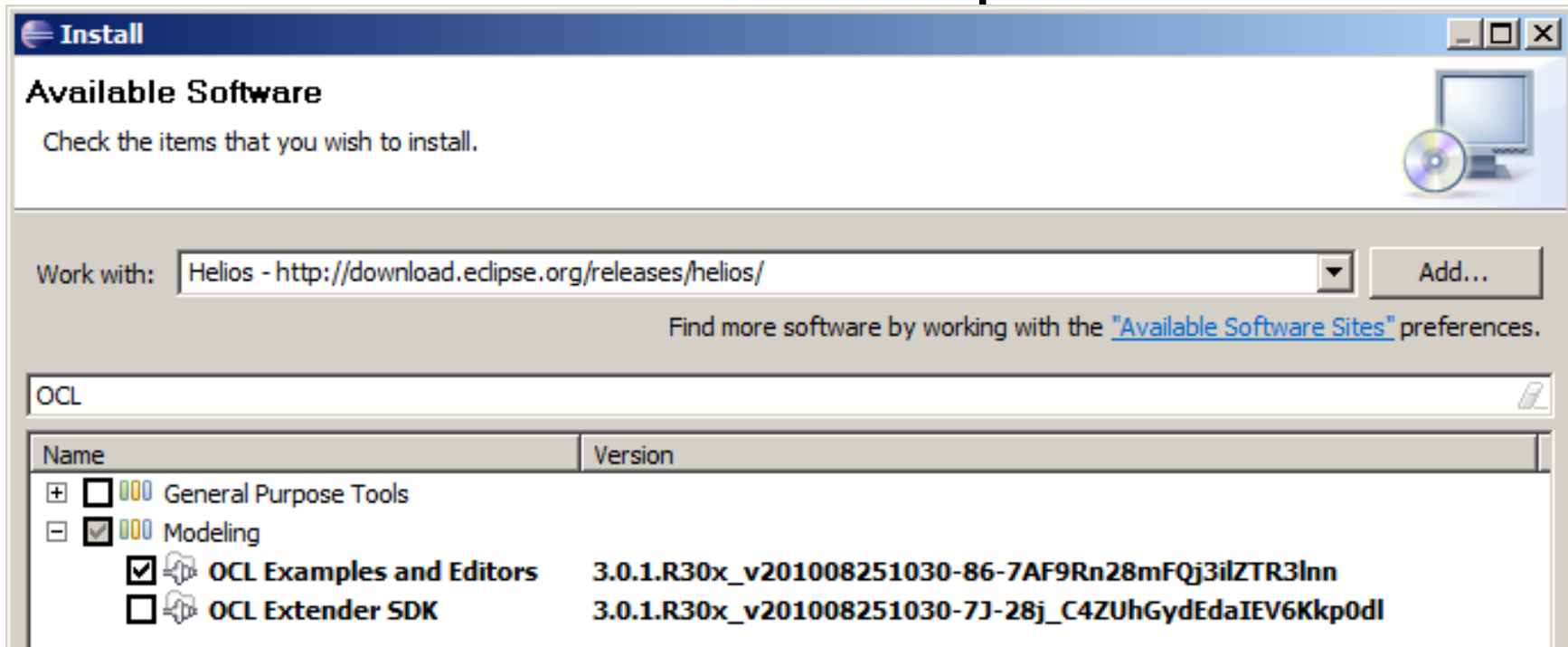
- Scaling up OCL at SAP

# Follow Along

<http://www.eclipsecon.org/summiteurope2010/sessions/?page=sessions&id=1710>

links to slides and to zip file comprising, model,edit,editor,diagram projects

- Install MDT/OCL 3.0.1 Examples and Editors



- Import ... Existing Projects from Archive
  - ESEExampleTree/model/People1.ecore
- Run nested Eclipse, Import ESEExampleTree
  - ESEExampleTree/model/default.people\_diagram

# How Much OCL?

## None

- Very simple modeling
- Java augmented modeling

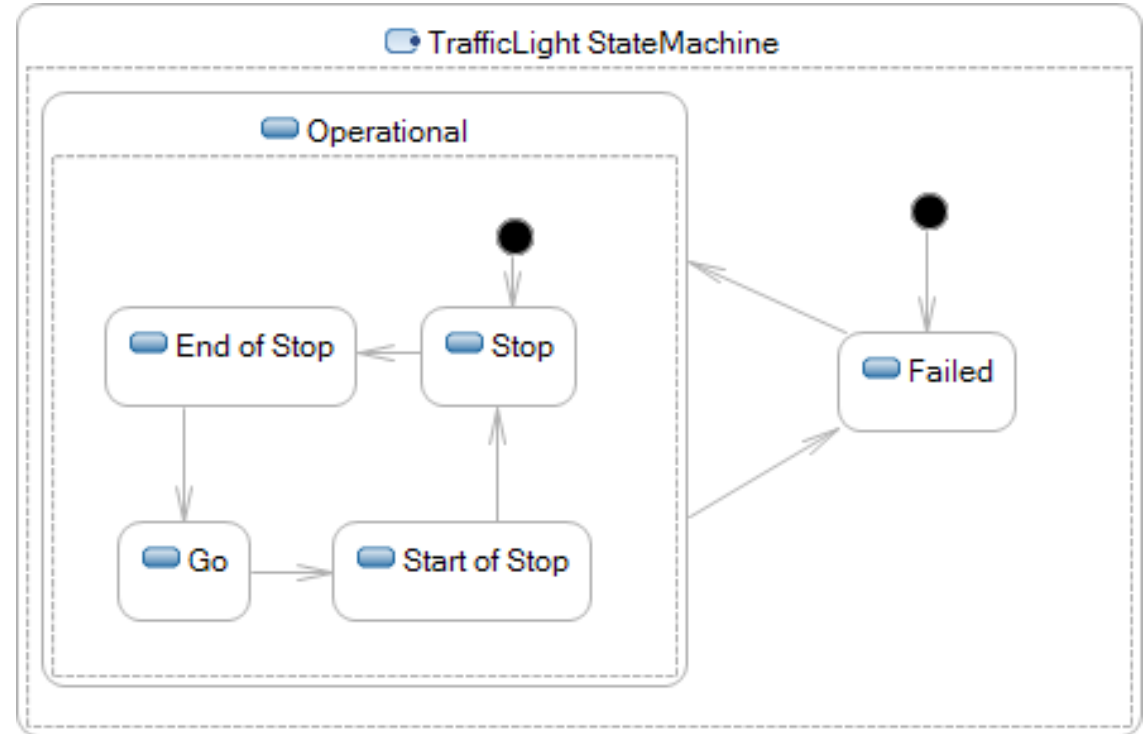
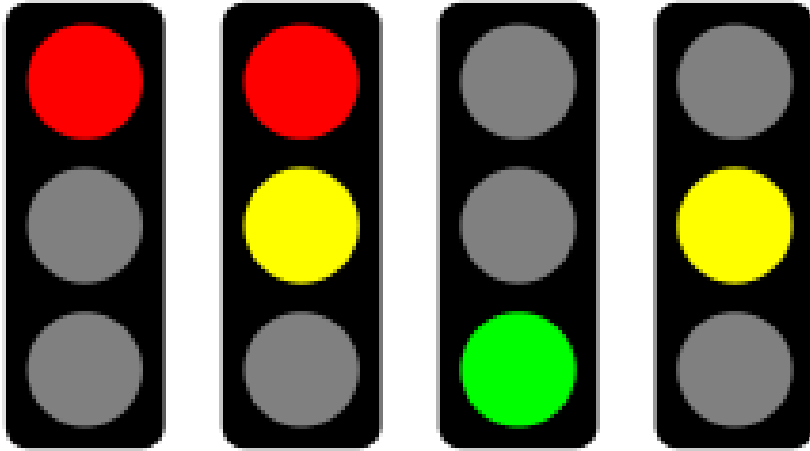
## A little

- OCL (and Java) augmented modeling

## A lot

- OCL as a powerful formal specification language
  - OMG's UML, OCL, QVT, ... specifications
- OCL as the foundation of a transformation language
  - MOFM2T (Acceleo), QVT
- OCL as a portable implementation language

# UML State Machines



- Need to specify behavior
  - amber when End of Stop or Start of Stop
  - transition when signal received/time elapsed

# UML Solutions

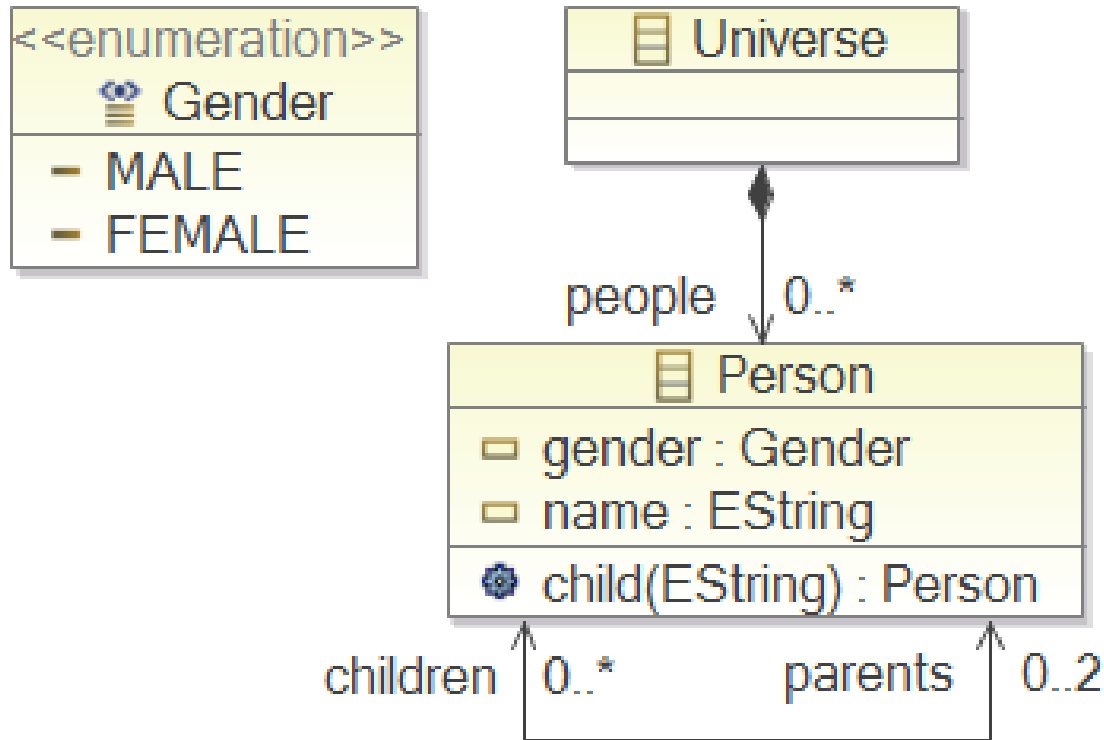
UML 1.x Use your favourite programming language

- Ada/C/...
- Magically inserted by proprietary code generator

UML 2.x Use a neutral specification language

- The Object Constraint Language
  - State machine guards/actions
  - Class invariants
  - Operation bodies, pre/post conditions
  - Property initial/derived values

# Simple Meta-Modeling



Example Family Tree Meta-Model  
Ecore Diagram  
(similar to UML Class Diagram)

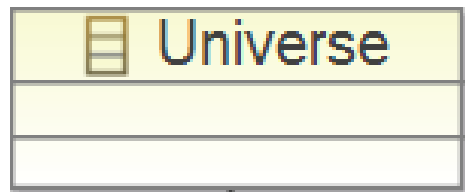
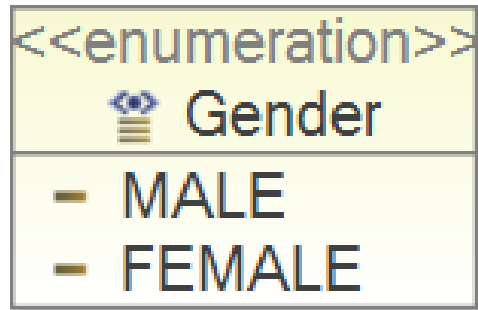
## Graphics

- Box
  - Class, enumeration
- Compartment
  - Property, operation
- Line
  - Association
- Decoration
  - Composition, navigability

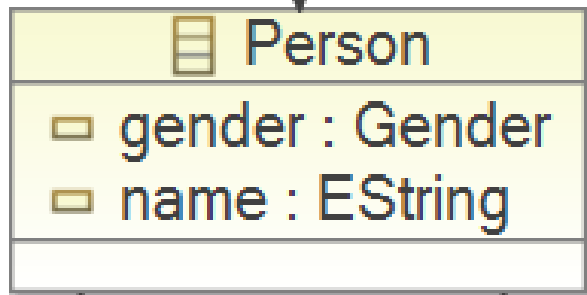
## Text

- Name, type, stereotype
- Multiplicity

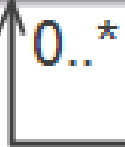
# Richer Meta-Modelling



people 0..\*



children 0..\*



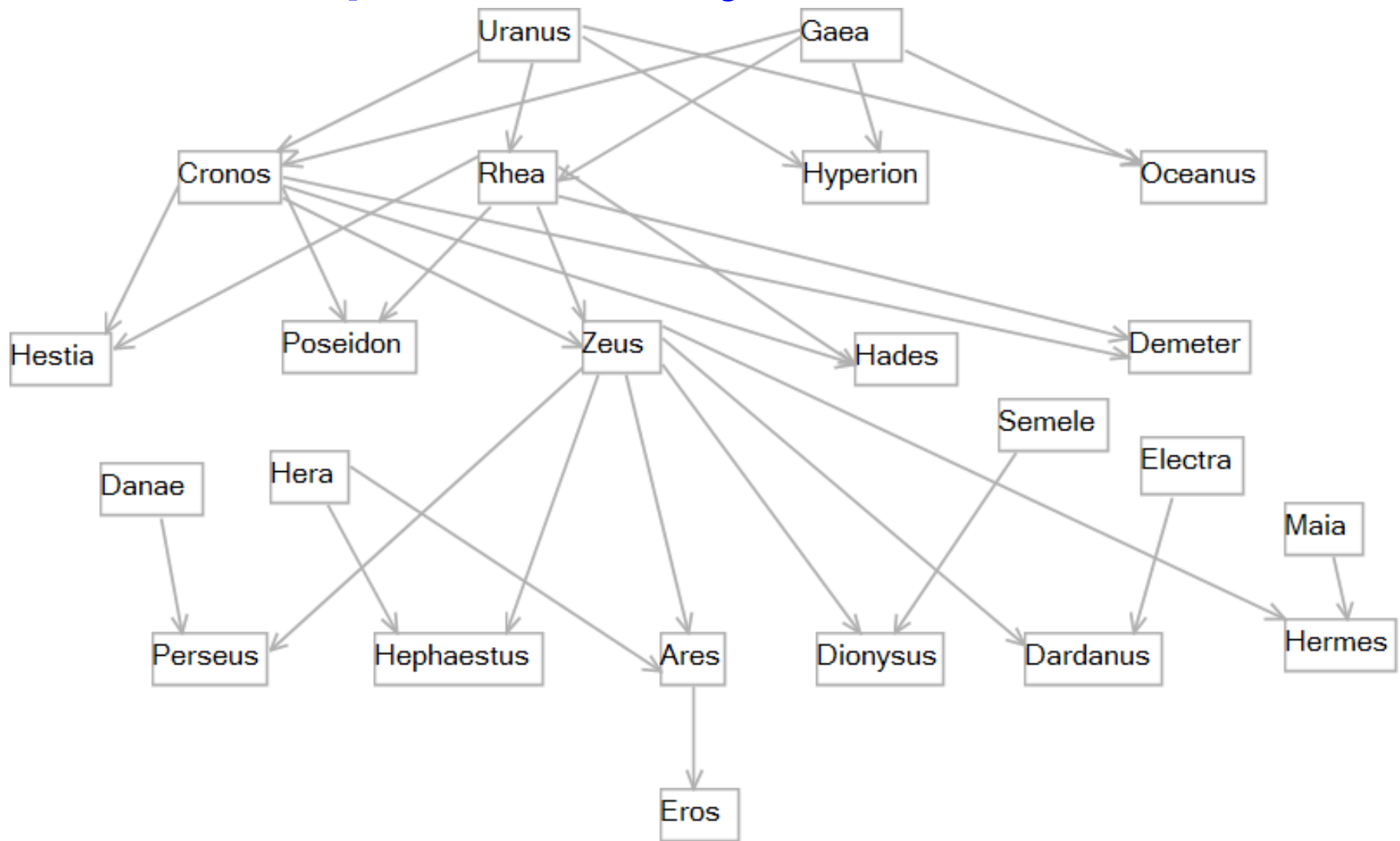
0..2

parents

- Implicit constraints
  - Up to 2 parents
  - MALE/FEMALE gender
- Arbitrary constraints
  - At least 5 characters in name
  - 1 MALE, 1 FEMALE parent
  - Self is not an ancestor of self



# Example Family Tree Model



## Simple GMF Diagram Editor

ESEExampleTree/model/default.people\_diagram

# Simple Query Evaluation

The screenshot displays the Eclipse IDE interface. On the left, a model diagram titled 'default.people\_diagram' shows a hierarchy of Greek gods. Zeus is selected as the model context. On the right, the 'Console' view is open, showing the 'Interactive OCL' console. The console displays the query 'children' and its results: 'Person Hephaestus', 'Person Ares', 'Person Dionysus', 'Person Perseus', 'Person Hermes', and 'Person Dardanus'. A red box highlights the 'children' text in the console, and a red arrow points to the 'New Console View' option in the 'Window->Show View->Other...' menu.

Window->Show View->Other... Console

Console: New: Interactive OCL

Select **Zeus** as the Model Context (in any model editor)

Type **children** then carriage return

The screenshot shows the 'Window->Show View->Other...' menu. The 'Interactive OCL' option is selected, indicated by a blue highlight. Other options visible include '1 Java Stack Trace Console', '2 Host OSGi Console', '3 CVS', and '4 New Console View'.

# OCL Principles

- Natural/Formal Language compromise
  - natural language error prone
  - formal language unapproachable to many
  
- Specification (not Programming) Language
  - declarative, modeling language
  - side effect free, no model changes, atomic execution
  - strongly typed, using UML generalization

# OCL Object Types

- Primitive Types

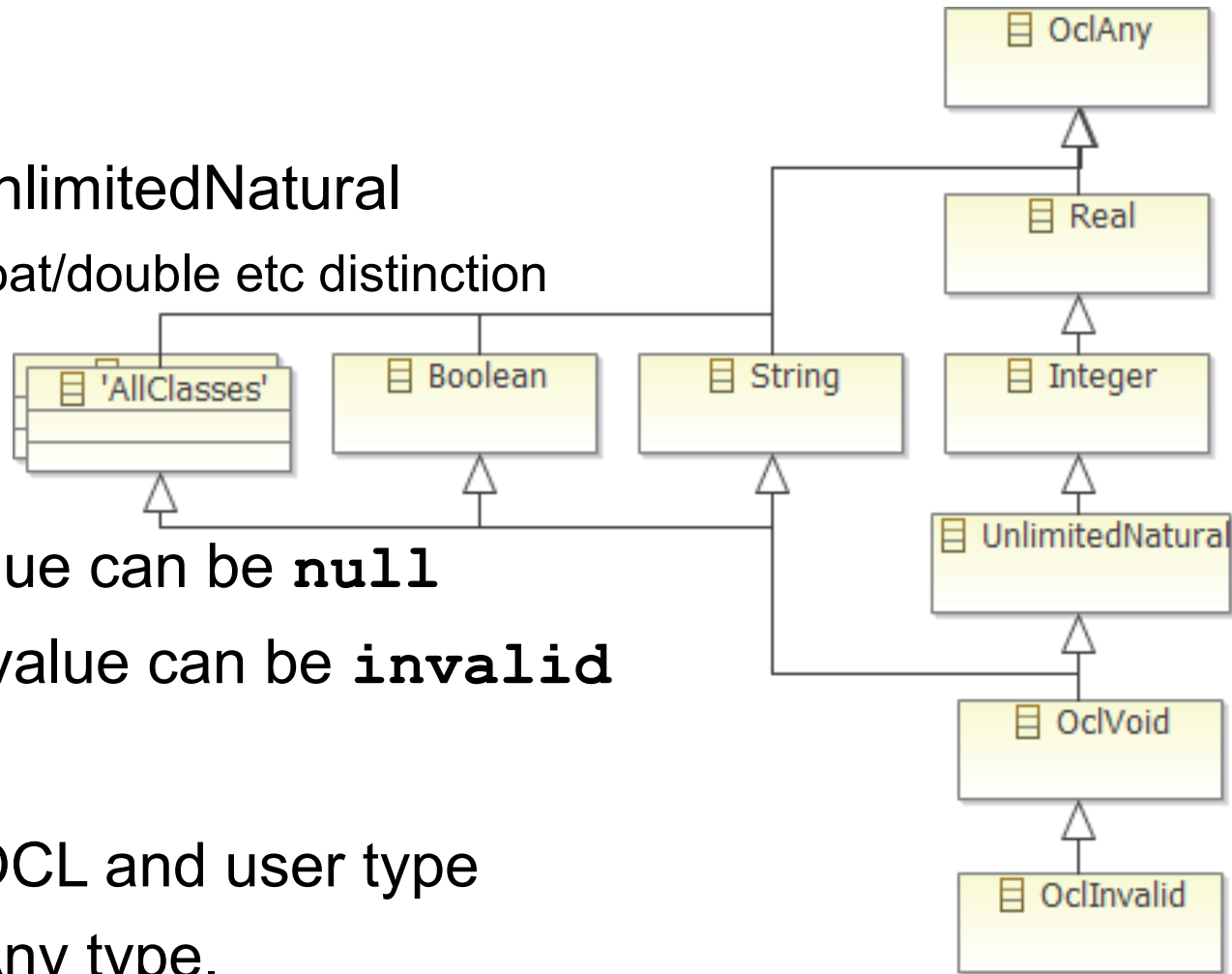
- Boolean, String
- Real, Integer, UnlimitedNatural
  - unlimited; no float/double etc distinction

- Bottom Types

- OclVoid: any value can be **null**
- OclInvalid: any value can be **invalid**

- Top Type

- OclAny: every OCL and user type conform to OclAny type.



# Mathematical Operators

Infix:            +, -, \*, /  
                  and, or, xor, implies  
                  =, <>, <, >, <=, >=                    nb =, <>

Prefix:          not, -

**4.0 \* -5**

**'a' + 'b'**

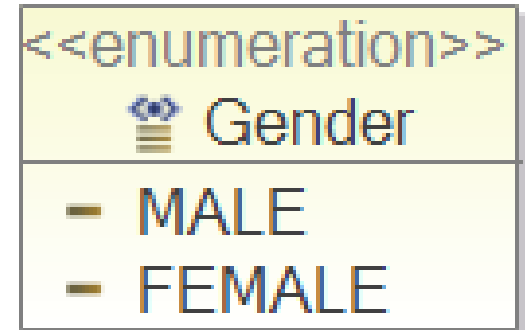
Operators: mod, div, max, min, ...

**4.max(5)**

# OCL Expressions

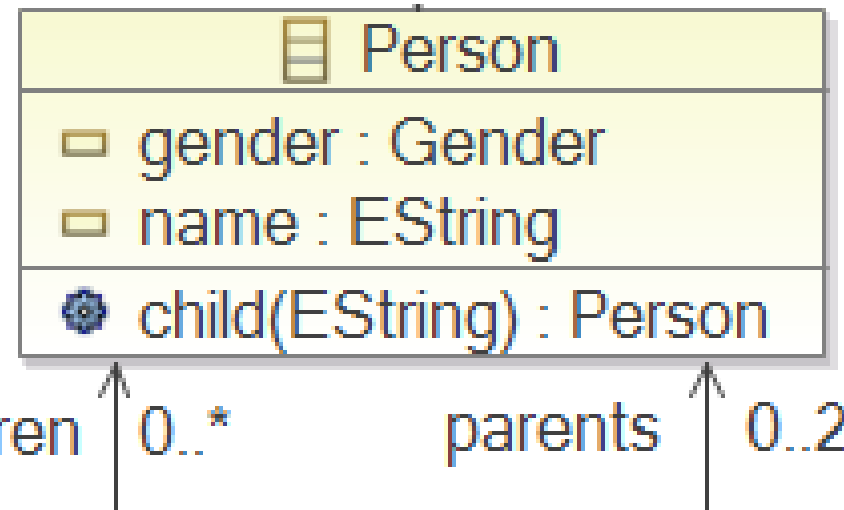
- If Expressions

```
if gender = Gender::MALE
then 'he'
else 'she'
endif
```



- Let Expressions

```
let jack : Person = child('Jack'),
    jill : Person = child('Jill')
in jack <> null and jill <> null
```



# More Complex Query

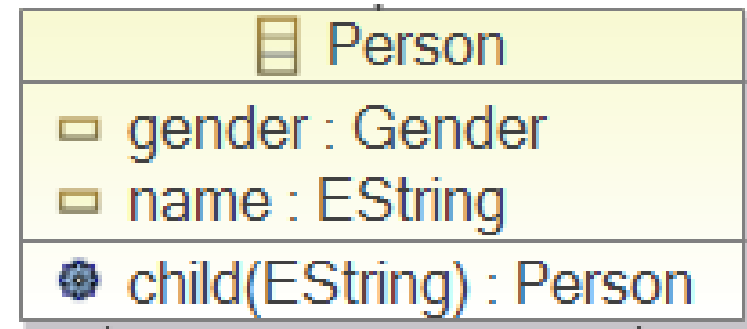
The screenshot displays the Eclipse IDE interface. On the left, a UML class diagram titled 'default.people\_diagram' shows a network of relationships between several entities: Cronos, Rhea, Poseidon, Zeus, Danae, and Hera. Poseidon is highlighted with a dashed border, indicating it is the selected element. On the right, the 'Console' window shows the 'Interactive OCL' environment. The console displays the following OCL query being evaluated:

```
let zero : Integer = 0 in
  if self.children->size() > zero
  then self.children->collect(name)
  else Sequence{'No children'}
  endif
```

The results of the evaluation are shown as 'No children'. A red box highlights the OCL code in the console.

Selecting *Poseidon* defines the implicit context variable  
**self : Person = Poseidon**

# Object Navigation



## Properties

- `self.name` or just `name`  
(cf. `this.getName()` or `getName()`)

## Operations

- `self.child('John')` or just `child('John')`  
(cf. `this.child('John')` or `child('John')`)



# The OCLinEcore Editor

```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore#/';  
  
package people : tree = 'http://www.eclipse.org/examples/tree'  
{  
    enum Gender  
    {  
        MALE;  
        FEMALE;  
    }  
    class Person  
    {  
        invariant AtLeastFiveLetters: name.size() >= 5;  
        property children#parents : Person[*];  
        property parents#children : Person[0..2];  
        attribute gender : Gender[1];  
        attribute name : String[1];  
    }  
    class Universe  
    {  
        property people : Person[*] { composes };  
    }  
}
```

Outline view structure:

- OCL in Ecore document
  - ecore :
    - people
      - Gender
      - Person
        - children : Person[\*]
        - parents : Person[0..2]
        - gender : Gender
        - name : String
      - <invariant> AtLeastFiveLetters
        - >=
        - .size
        - 5
    - Universe

Context menu options:

- New
- Open (F3)
- Open With (selected)
- Show In (Alt+Shift+W)
- Copy (Ctrl+C)

Sub-menu for 'Open With':

- OCLinEcore (Ecore) Editor (selected)
- Sample Ecore Model Editor
- Sample Reflective Ecore Model Editor
- Text Editor

Select model/People1.ecore

Open With... OCLinEcore

# Example Validation Failure

The screenshot shows the Eclipse IDE interface. On the left, a tree view displays a model structure under 'platform:/resource/ESEExampleTree/model/People1.xmi'. The 'Universe' element is selected, and a right-click context menu is open with the 'Validate' option highlighted. On the right, a 'Validation Problems' dialog box is displayed. The dialog has a title bar with a close button and a main area with a red 'X' icon and the text 'Problems encountered during validation'. Below this, it says 'Reason: Diagnosis of Universe'. At the bottom of the dialog, there are 'OK' and '<< Details' buttons. A list of validation errors is shown in a scrollable area, each preceded by a red 'X' icon:

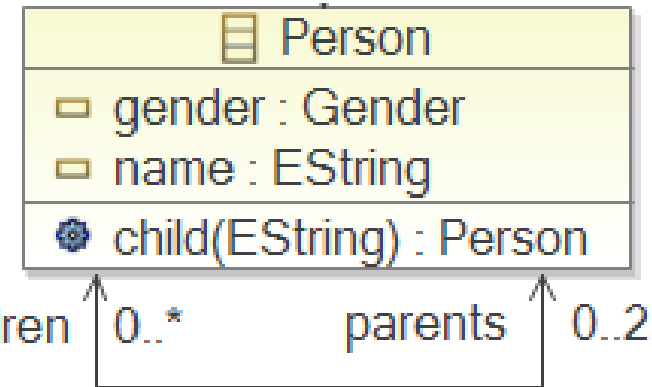
- ✗ The 'AtLeastFiveLetters' constraint is violated on 'Person Zeus'
- ✗ The 'AtLeastFiveLetters' constraint is violated on 'Person Hera'
- ✗ The 'AtLeastFiveLetters' constraint is violated on 'Person Ares'
- ✗ The 'AtLeastFiveLetters' constraint is violated on 'Person Eros'
- ✗ The 'AtLeastFiveLetters' constraint is violated on 'Person Rhea'
- ✗ The 'AtLeastFiveLetters' constraint is violated on 'Person Gaea'
- ✗ The 'AtLeastFiveLetters' constraint is violated on 'Person Maia'

Open model/People1.xmi with Sample Reflective Ecore Editor  
Select Universe, Right button menu, Validate

# Multiplicities and Collections

## Meta-models specify multiplicities

- children : Person[\*] {ordered,unique}
- parents : Person[0..2] {unique}



- multiplicities are specification concepts; not objects

## Implementations (e.g. Ecore) reify multiplicities

- `getChildren()` returns a `UniqueEList<Person>`
- 'many' properties have extra implementation objects
  - `getName()`                      `setName(newName)`
  - `getChildren().get(2)`          `getChildren().add(newChild)`

## OCL needs more than just UML multiplicities

# OCL 2.0 Collections

Typed Collections partially reify multiplicities

Collection(T)	Unordered	Ordered
Non-Unique	Bag(T)	Sequence(T)
Unique	Set(T)	OrderedSet(T)

Collections are different to objects

Navigation from a Collection uses ->

- [Navigation from an Object (OclAny) uses .]

Collections have type parameters

Collections have useful operations

Collections have very useful iterations

# Example Collection Operations

Collection::size()      `self.children->size()`

'get'

Sequence::at(Integer)      `self.children->at(1)`

– nb 1 is the first index, `size()` is the last

'add'

Collection(T)::including(T) : Collection(T)

– returns a new collection with added content

'contains'

Collection(T)::includes(T) : Boolean

– tests existing content

# Collection::select iteration

- Children

```
self.children
```

- Sons

```
self.children->select(gender = Gender::MALE)
```

```
self.children->select(child | child.gender = Gender::MALE)
```

```
self.children->select(child : Person | child.gender = Gender::MALE)
```

- `select(iterator : type | body)`
  - filters to select elements for which the body is true
- `reject(iterator : type | body)`
  - filters to reject elements for which the body is true
- cf multi-line Java loop

# Collection::collect iteration

- Children

```
self.children
```

- Grandchildren

```
self.children->collect(children)
```

```
self.children->collect(child | child.children)
```

```
self.children->collect(child : Person | child.children)
```

- collect(iterator : type | body)

- creates a new collection comprising all the bodies

- any, exists, forAll, isUnique, iterate, one,

# OCL Navigation Operators

`anObject.` ... object navigation  
`aCollection->` ... collection navigation

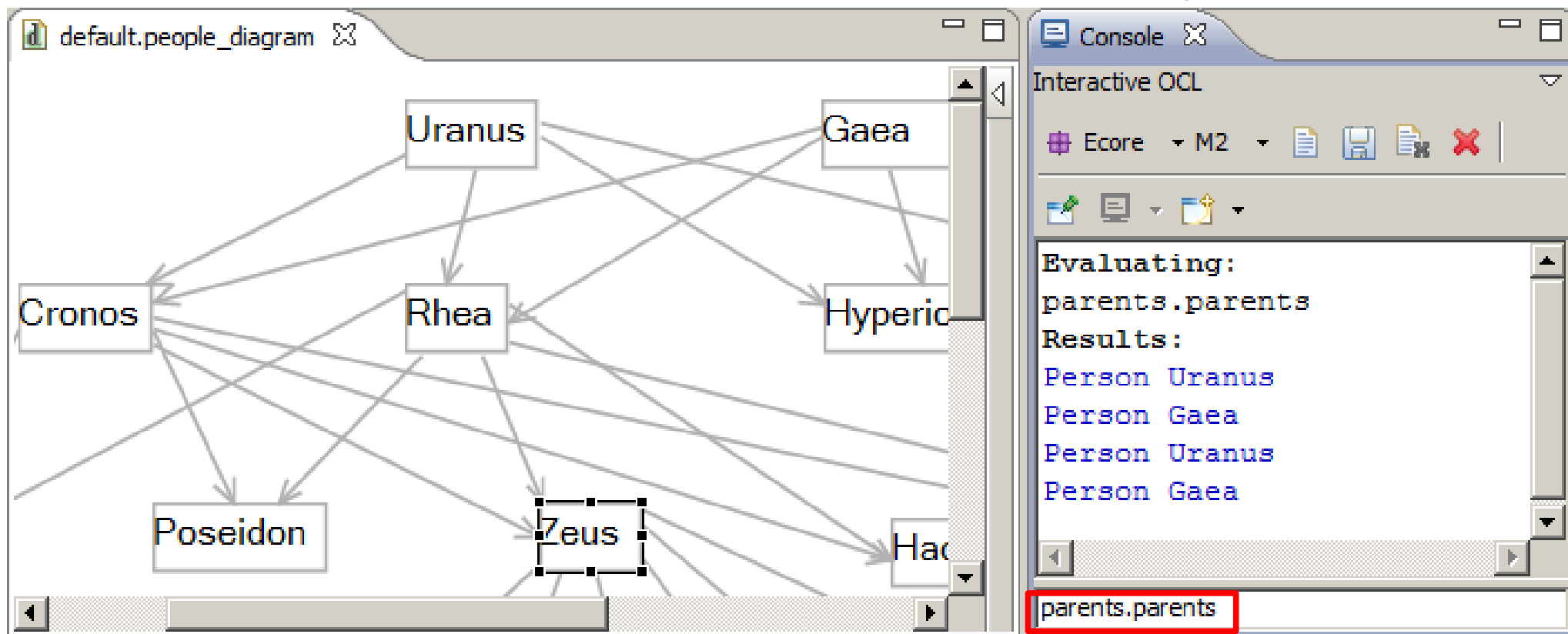
	Object	Collection
.	Navigation	?
->	?	Navigation

## Shorthands

`aCollection.` ... implicit collect  
`anObject->` ... implicit collection



# Implicit Collect Query

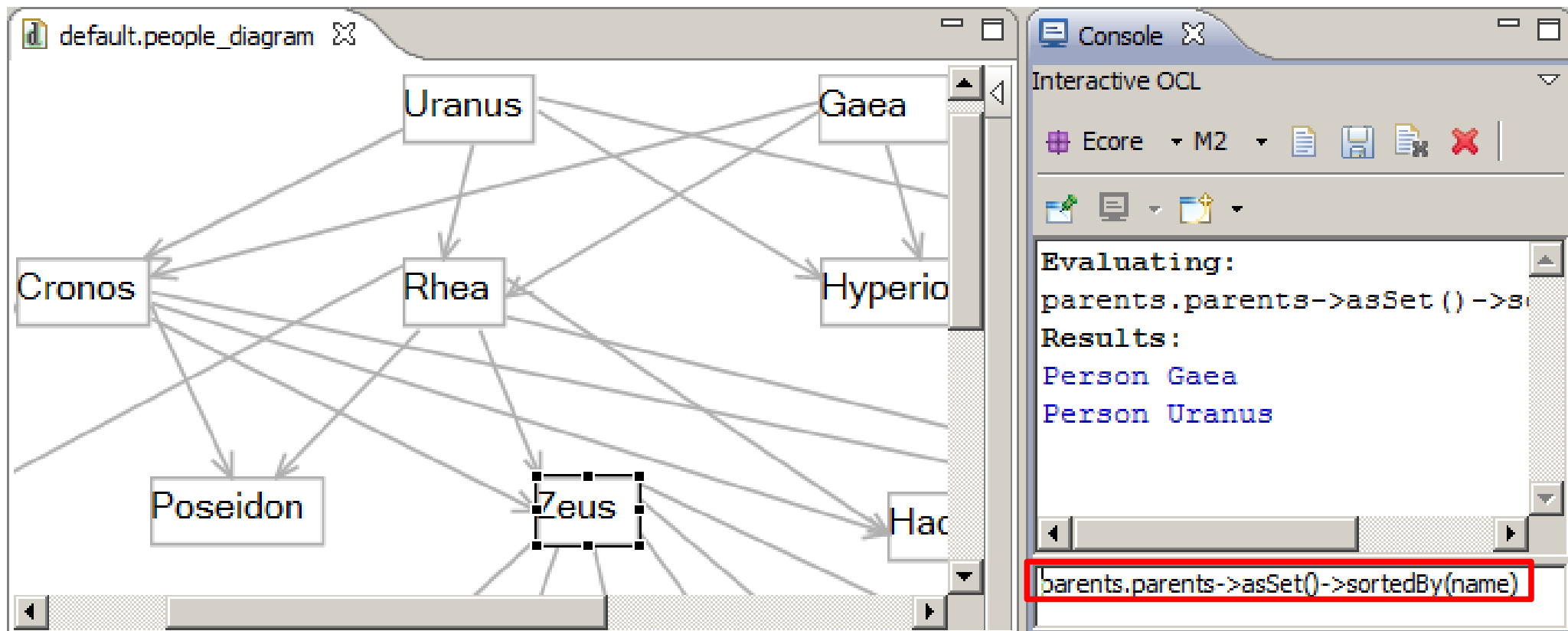


**parents.parents = parents->collect (parents)**

3 symbols, compared to 4 lines of Java

4 grandparents, but not all different!

# Cleaned up query



`parents.parents->asSet() ->sortedBy(name)`

`->asSet()` converts to `Set(Person)`, removes duplicates

`->sortedBy(name)` alphabeticizes

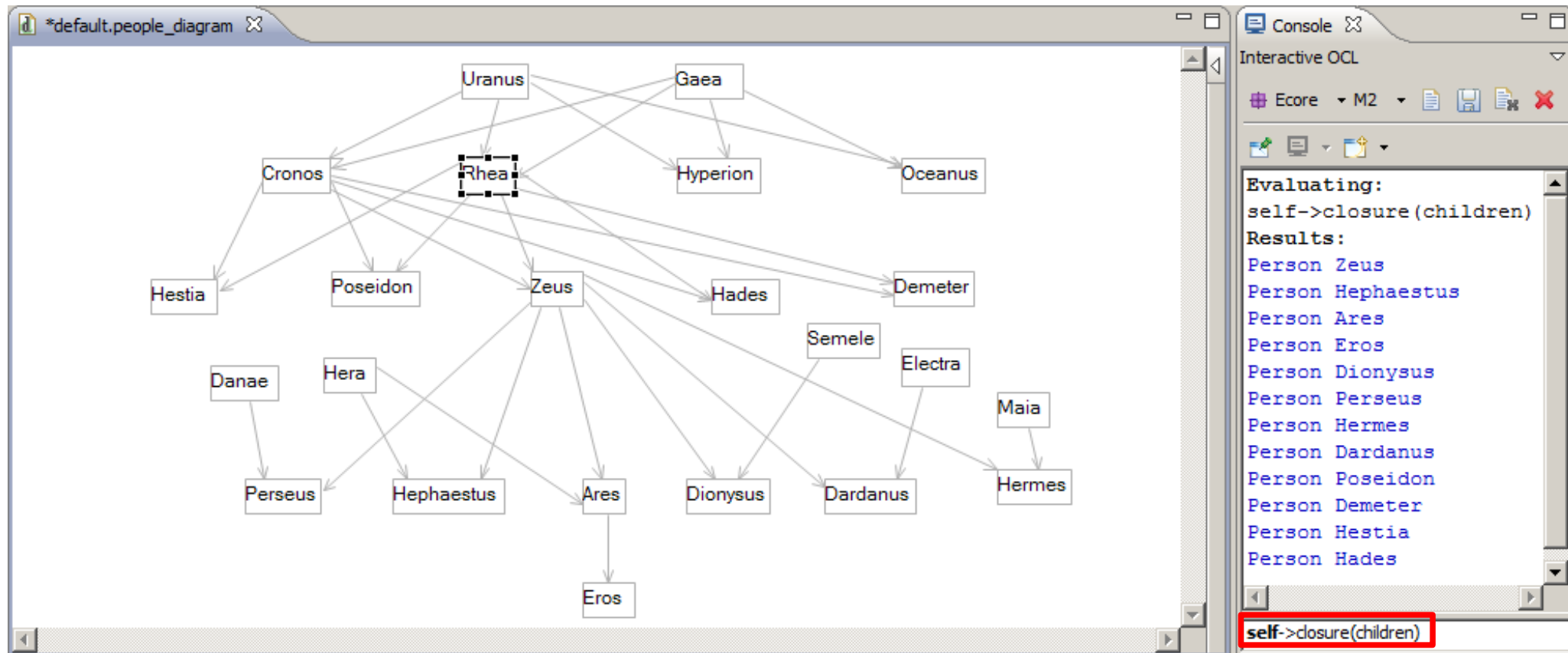
# Implicit Collection Conversion

	Object	Collection
.	Navigation	Implicit collect()
->	Implicit Collection	Navigation

## self->notEmpty()

- Standard OCL idiom
  - Converts self (if an object) to a Collection of self
  - If self is a defined object
    - Implicit collection is not empty - true
  - If self is an undefined object (null)
    - Implicit collection is empty - false
  - If self is an error (invalid)
    - Implicit collection is also an error - invalid

# Collection::closure iteration



- children, grandchildren, greatgrandchildren etc  
`self->closure(children)`
- Implicit collection of self, then closure of all children  
[ closure in MDT/OCL 1.2, probably in OMG OCL 2.3 ]

# OCL as Implementation

```
class Person
{
    invariant AtLeastFiveLetters: name.size() >= 5;
    invariant MixedGenderParents: father <> null and mother <> null;
    invariant SelfIsNotAncestorOfSelf: self->closure(parents)->excludes(self);
    property children#parents : Person[*];
    property parents#children : Person[0..2];
    attribute gender : Gender[1];
    attribute name : String[1];
    property father : Person[1] { derived,transient,volatile }
    {
        derivation: parents->any(c : Person | c.gender = Gender::MALE);
    }
    property mother : Person[1] { derived,transient,volatile }
    {
        derivation: parents->any(c : Person | c.gender = Gender::FEMALE);
    }
    operation child(childName : String) : Person
    {
        body: children->any(c : Person | c.name=childName);
    }
}
```

**any (x)** iteration selects an arbitrary element for which x is true.

# Derived Properties

The screenshot shows the Eclipse IDE interface. On the left, the 'People3.xmi' file is open, displaying a tree view of the model. The 'Universe' container is expanded, showing several 'Person' objects: Zeus, Hera, Hephaestus, Ares, Eros, Cronos, and Rhea. The 'Person Zeus' object is selected. On the right, the 'Properties' view displays the properties of the selected object. The table below represents the data shown in the Properties view.

Property	Value
Children	Person Hephaestus, Person Ares, Person Dionysus, Person Perseus, Person Hermes,
Father	Person Cronos
Gender	MALE
Mother	Person Rhea
Name	Zeus
Parents	Person Cronos, Person Rhea

The screenshot shows a 'Validation Problems' dialog box. The title bar reads 'Validation Problems'. The main area contains a red 'X' icon and the text 'Problems encountered during validation'. Below this, the 'Reason:' is 'Diagnosis of Person Hera'. At the bottom, there are 'OK' and '<< Details' buttons. A list of validation errors is shown at the bottom of the dialog:

- ✗ The 'AtLeastFiveLetters' constraint is violated on 'Person Hera'
- ✗ The 'MixedGenderParents' constraint is violated on 'Person Hera'
- ✗ The required feature 'father' of 'Person Hera' must be set
- ✗ The required feature 'mother' of 'Person Hera' must be set

## For Hera

```
invariant MixedGenderParents:  
father.gender <>  
mother.gender;
```

fails because  
father is **null** and  
mother is **null**

# Other OCL Capabilities

No time to mention

- Other iterators, operations
- Tuples
- Association Classes/Qualifiers
- @pre values
- Messages
- States

# OMG OCL Progress

## OCL 2.2 (current) Collections are objects!

- Collection conforms to OclAny
- No need for Collection/Object polymorphic operations
- Collections can mix Object/Collection content

## ? OCL 2.4 Specification defined by models

- Auto-generated by Acceleo
- Fix too many consistency/typo/realizability issues
- Aligned with UML 2.4, MOF 2.4, XMI 2.4

Eclipse committers active on OMG RTF



# Eclipse MDT/OCL

OMG OCL 1.x  
EMFT/OCL 1.0

- Original code contribution by IBM
- Java callable API
  - Parse/evaluate OCL 1.x against Ecore meta-models

OMG OCL 2.0  
MDT/OCL 1.2

- Ecore or UML meta-models
- OCL 2.0 (in so far as possible)
- Example Interactive Console

OMG OCL 2.2  
MDT/OCL 3.0

- towards OCL 2.2
- Example Xtext editors (Ecore only)

# Validation History

Use of OCL to define model validation

Eclipse 3.2

- EMF Validation Framework
  - Embed OCL in XML CDATA

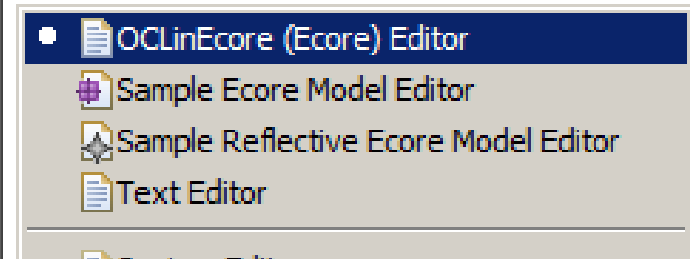
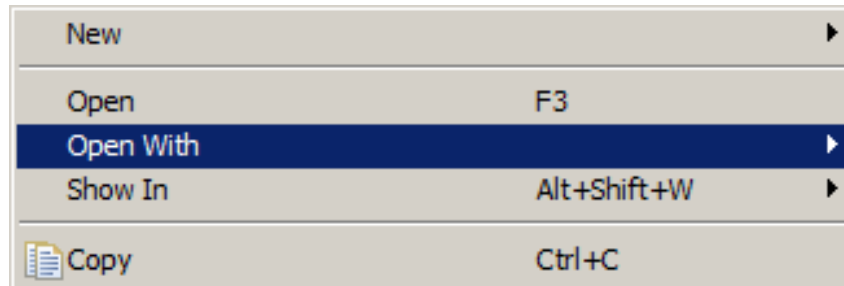
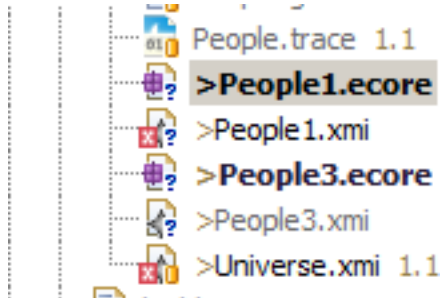
Eclipse 3.3

- EMF, OCL EAnnotations
  - Embed OCL in EAnnotation
  - Genmodel to integrate

Eclipse 3.6  
EMF 2.6  
MDT/OCL 3.0

- Delegates
  - Embed OCL in EAnnotation
  - EObject.eInvoke() for dynamic invocation
  - OCLinEcore editor for semi-validated editing

# OCLinEcore Editor

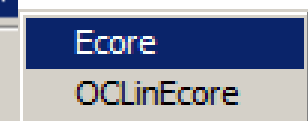
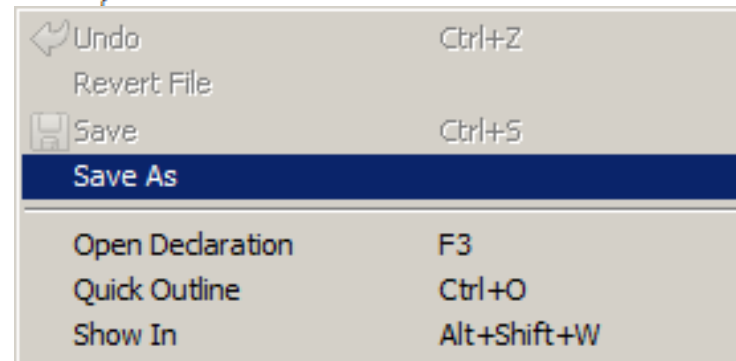


- Open with -> OCLinEcore
- Save As \*.ecore
  - Loses formatting and comments
- Save As \*.oclinecore
  - Text file preserves comments
- Useful for plain Ecore too:
  - Printable/reviewable text
  - Searchable/replaceable text

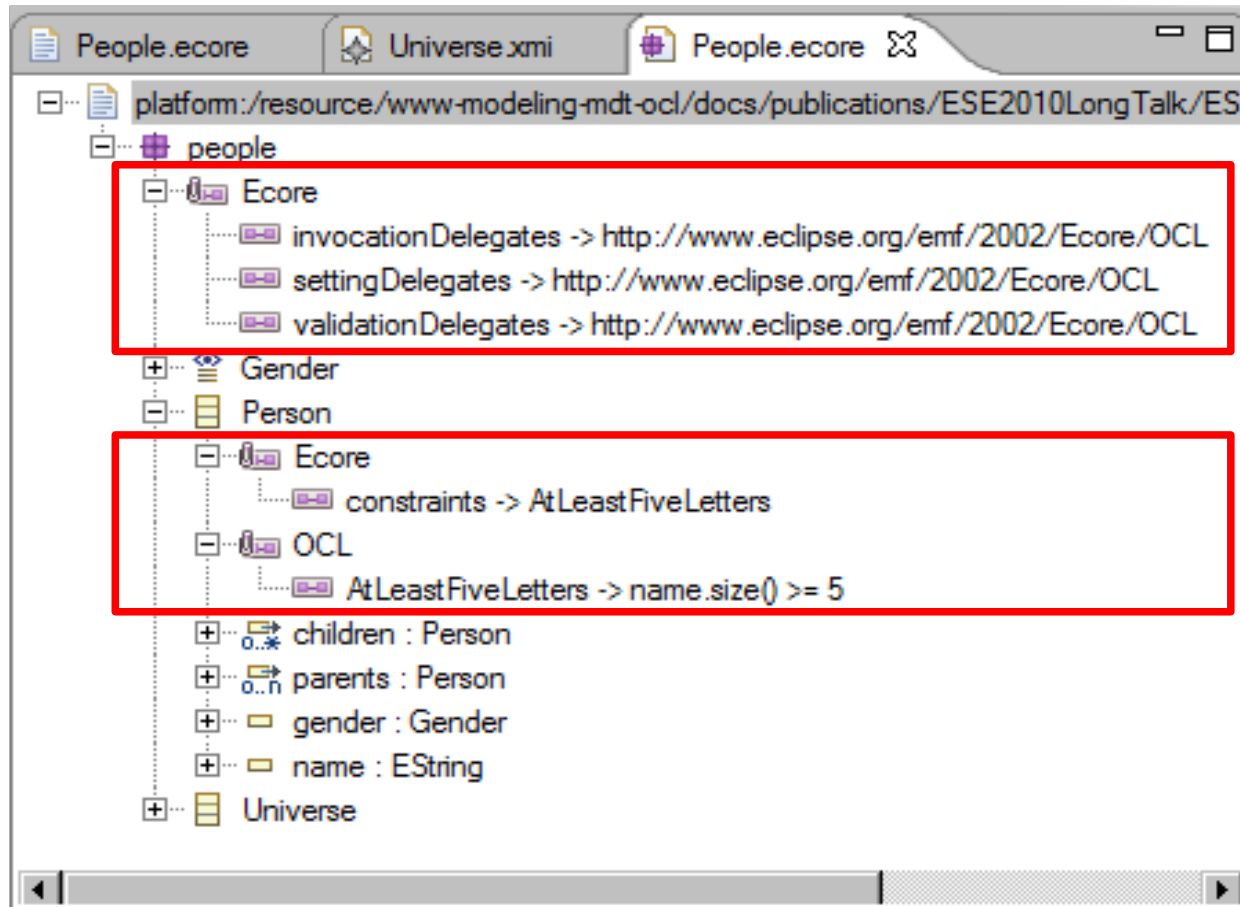
```

package people : tree = 'http://www.eclipse.org/examples/tree'
{
    enum Gender
    {
        MALE;
        FEMALE;
    }
    class Person
    {
        invariant AtLeastFiveLetters: name.size() >= 5;
        property children#parents : Person[*];
        property parents#children : Person[0..2];
        attribute gender : Gender[1];
        attribute name : String[1];
    }
}

```



# Validation in Sample Ecore Editor



OCLinEcore editor maintains EAnnotations automatically  
OCLinEcore editor provides OCL syntax checking  
OCLinEcore editor will provide OCL semantic checking

# (Example) Tools and Tips

OCLEcore editor for Ecore/embedded OCL

CompleteOCL editor for OCL documents

EssentialOCL editor for OCL Expressions (Papyrus)

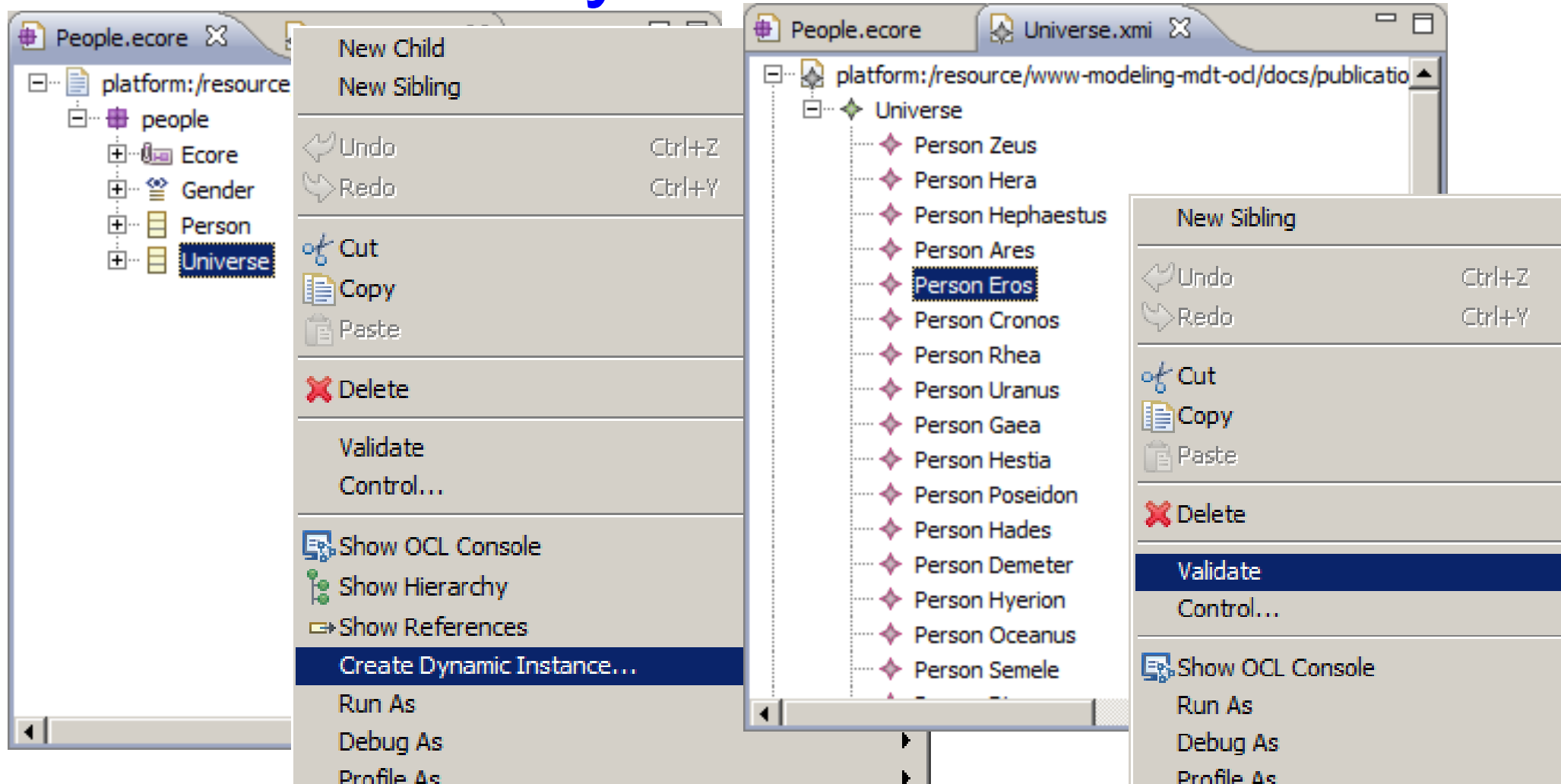
OCL Interactive Console

- Invaluable ability to practice non-trivial expressions
- Page Up/Page Down to reuse expressions

Meta-model reload after change

Genmodel settings for embedded OCL

# EMF Dynamic Instances

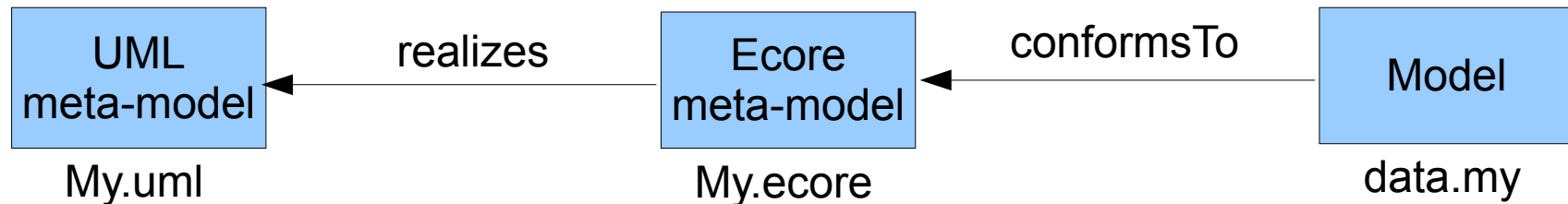


Create/update Ecore meta-model

Create XMI instance of EClass in meta-model

Update XMI model, validate OCL constraints

# Meta-model Update



## Edit UML/Ecore meta-model in UML/Ecore editor

- manual export of UML to Ecore in workspace
- manual save of Ecore to workspace

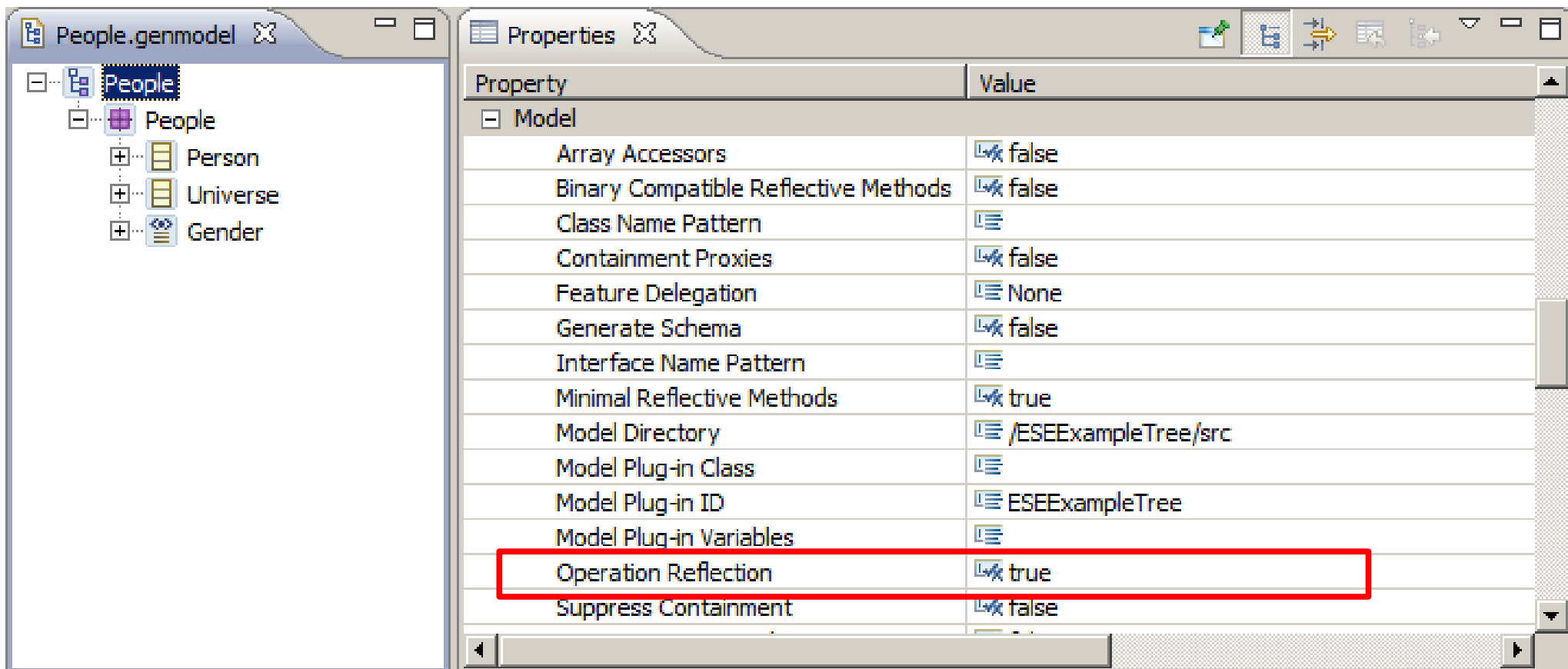
## Create Dynamic Instance/Load Model in editor

- validate/evaluate OCL constraints

## EMF does not support meta-model mutation

- Model.eClass() reverts to an unresolved proxy
- must exit and re-enter model editor

# Genmodel settings for OCL



Property	Value
Model	
Array Accessors	<input checked="" type="checkbox"/> false
Binary Compatible Reflective Methods	<input checked="" type="checkbox"/> false
Class Name Pattern	<input type="text"/>
Containment Proxies	<input checked="" type="checkbox"/> false
Feature Delegation	<input type="text"/> None
Generate Schema	<input checked="" type="checkbox"/> false
Interface Name Pattern	<input type="text"/>
Minimal Reflective Methods	<input checked="" type="checkbox"/> true
Model Directory	<input type="text"/> /ESEExampleTree/src
Model Plug-in Class	<input type="text"/>
Model Plug-in ID	<input type="text"/> ESEExampleTree
Model Plug-in Variables	<input type="text"/>
Operation Reflection	<input checked="" type="checkbox"/> true
Suppress Containment	<input checked="" type="checkbox"/> false

If not set to true

- MDT/OCL 3.0.0 OCL operation bodies not invoked
- MDT/OCL 3.0.1 Error Log as dynamic fallback used



# Eclipse MDT/OCL Futures

## 3.1 Core (Indigo)

- Minor maintenance

## 3.1 Examples (Indigo)

- New Ecore/UML blind pivot meta-model
- Extensible modelled Standard Library
- Xtext editors
- Super-compliant - anticipating OMG OCL resolutions

## 4.0 Core + Tools + Examples (Indigo+1)

- 3.1 Examples promoted to Core or Tools
  - preserved external APIs, significant revision of internal APIs
- OCL to Java code generation

# Which OCL Use Cases work When

	Validate	Evaluate	Console	Editor
Static Java For Ecore	1.0	1.0	1.0 Examples	3.0 Examples
Static Java For UML	1.2	1.2	3.1 Examples	3.1 Examples
Complete OCL For Ecore	3.1 Examples	3.1 Examples	3.1 Examples	3.0 Examples
Complete OCL For UML	3.1 Examples	3.1 Examples	3.1 Examples	3.1 Examples
Embedded OCL in Ecore	3.0	3.0	3.0 Examples	3.0 Examples
Embedded OCL in UML	3.1 Examples	3.1 Examples	3.1 Examples	3.1 Examples

Released in Helios Example functionality in Helios  
 Example functionality in Indigo, release in Indigo+1

# OCL 'Standard' Library

## Problems: OMG

- library is not a model
- uses non-UML concepts (Iterator)
- no reflection for OclAny::oclType()

## Problems: MDT/OCL

- hard coded, difficult to extend
- UML/Ecore differences, long generic template lists
- Ecore/EMOF discrepancies : EClass/Class

## Solution: OMG

- library is a model defined by the new OCL meta-model

## Benefit: MDT/OCL

- variants, extensible, unified, compliant

# OCL Models

## Problems: OMG

- OCL is not fully UML-aligned
- OCL modifies UML models (OclAny)
- Complete OCL modifies UML models
- OCL requires a modified sub-UML @ run-time

## Problems: MDT/OCL

- UML/Ecore implementation differences, Ecore extension
- Ecore/EMOF discrepancies

## Solution: OMG

- Pivot meta-model defines UML @ run-time
- Pivot model realises OCL-defined merges

## Benefit: MDT/OCL

- unified, compliant, Ecore/EMOF hidden

# Evaluation

## Problems: MDT/OCL:

- OCL interpreted by Java
- OperationCallExp visit is very inefficient
- Slightly hard to extend for QVTo, Acceleo
- OCL within genmodelled Java is just a String
  - significant first time parsing costs

## Solution: MDT/OCL

- OCL to Java code generation
- Library model references a Java class per feature
- Code efficiency

## Benefit: MDT/OCL

- extensible, faster (10 to 100 times ... iteration strategies)
- Java in genmodelled Java

# Beyond OCL

OMG OCL is a powerful expression language

- Declarative, First Order Predicate Calculus/Logic
- Model-oriented, UML navigation, multiplicities, ...

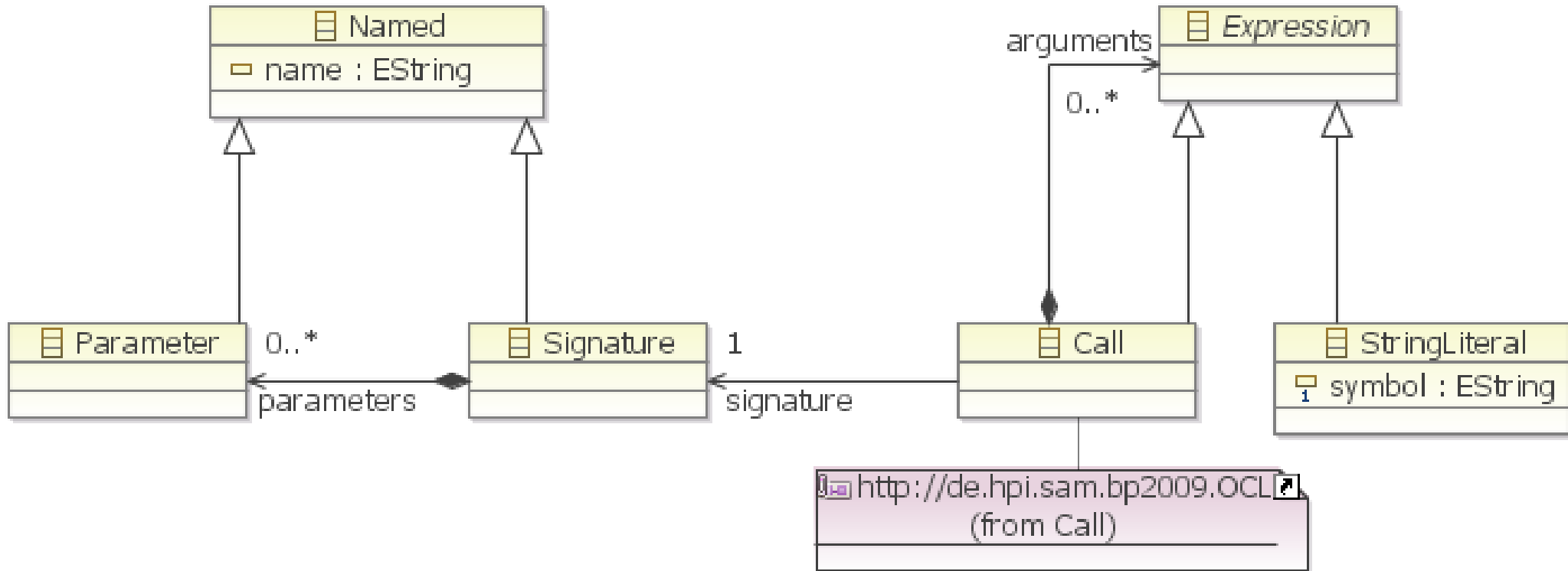
Formal language supports formal analysis  
analysis supports optimisation

OCL's usefulness calls for scalable  
implementation

# The Re-Evaluation Problem

- A set of OCL expressions
- A set of model elements
- A model change notification
- Which of the OCL expressions may have changed its value on which context elements?
- Naïve approach
  - re-evaluate all expressions for all their contexts
  - takes  $O(|\text{expressions}| * |\text{modelElements}|)$

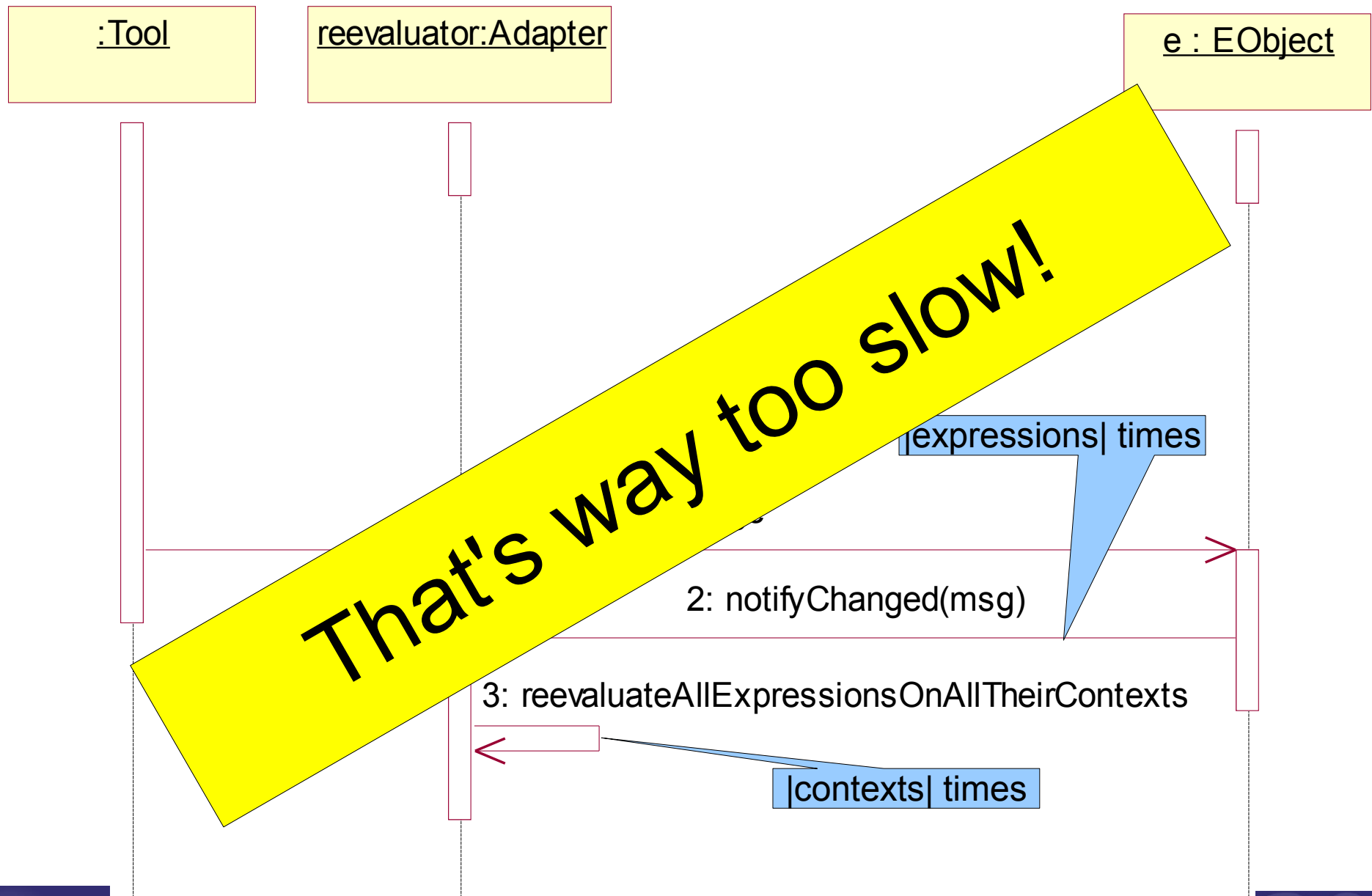
# Example



```
self.signature.parameters->size() = self.arguments->size()
```



# Naïve Re-Evaluation Sequence



# Idea: Find out from Notification which OCLExpressions may have changed

## Example: OCLExpression

```
self.arguments->size() = self.signature.parameters->size()
```

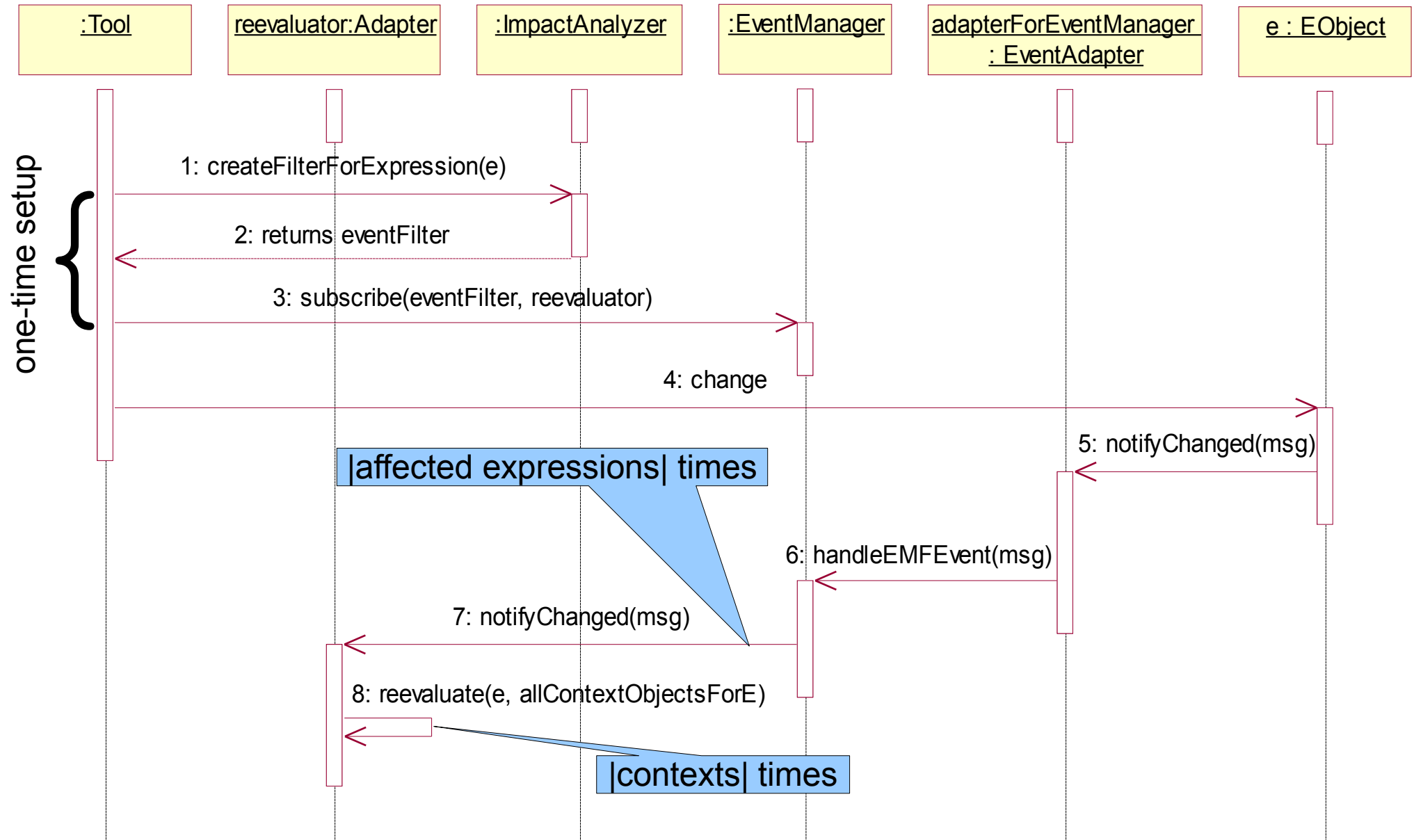
## generates **Notification** filter

```
(containment AND ((new value filter incl subs for: Call) OR  
                  (old value filter incl subs for: Call))) OR  
(wantedClass conformsTo: Signature) AND (feature: parameters)) OR  
(wantedClass conformsTo: Call) AND (feature: signature)) OR  
(wantedClass conformsTo: Call) AND (feature: arguments))
```

## Many expressions cause

- many adapters
- with one (often non-trivial) **Notification** filter each
- which need evaluation for each change **Notification**

# Filter Events for OCLExpressions

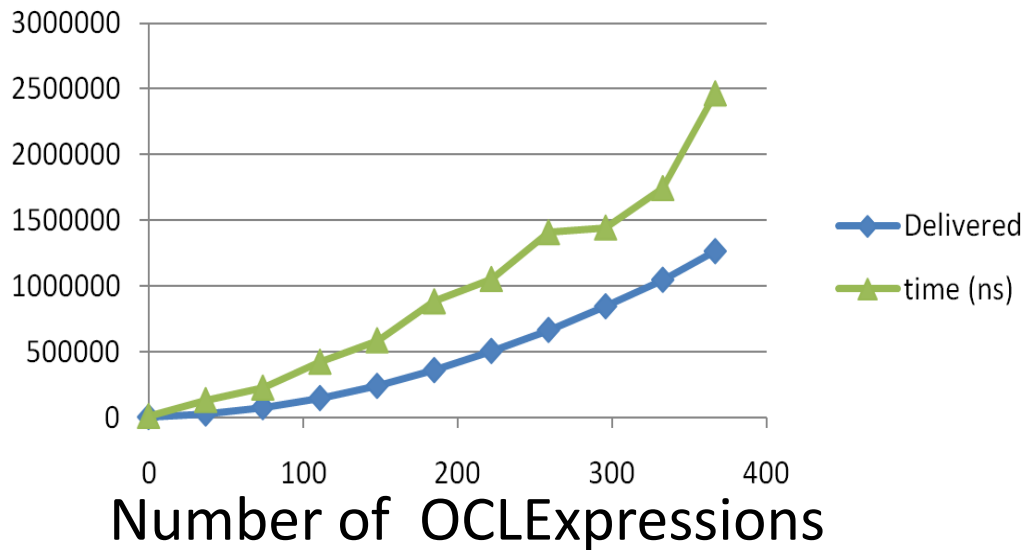


# Scaling up Event Filtering

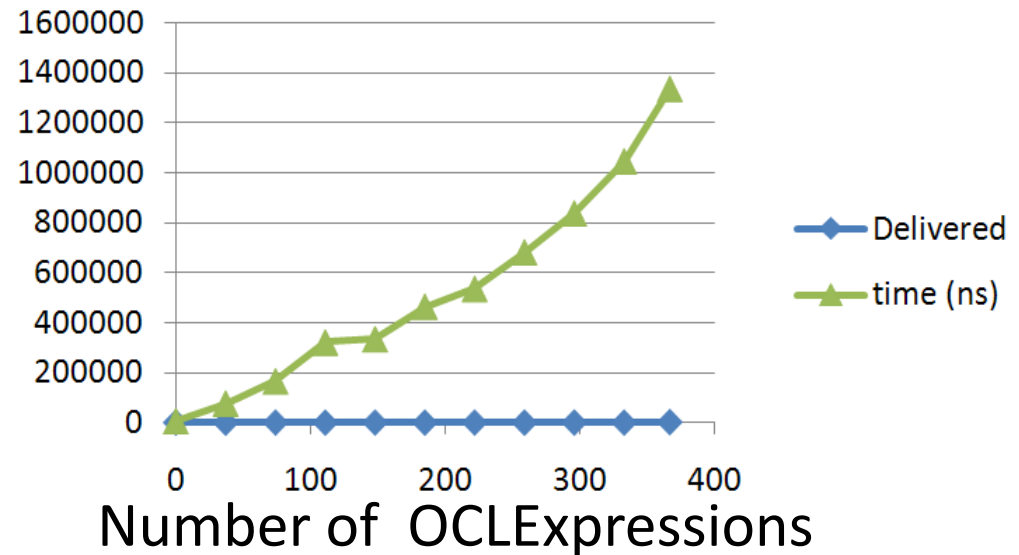
Effort for event propagation still  $O(|\text{expressions}|)$

- slowed down even if no **Notification** delivered

Notification #1



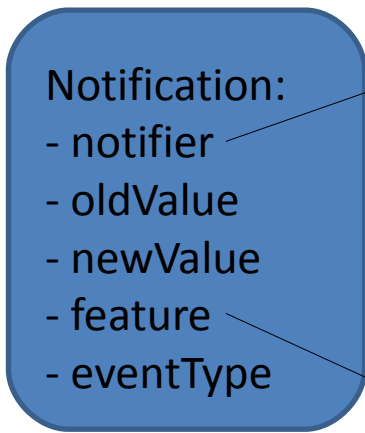
Notification #2



# Idea: Use HashMaps



to map `Notification` to `Set<Adapter>`



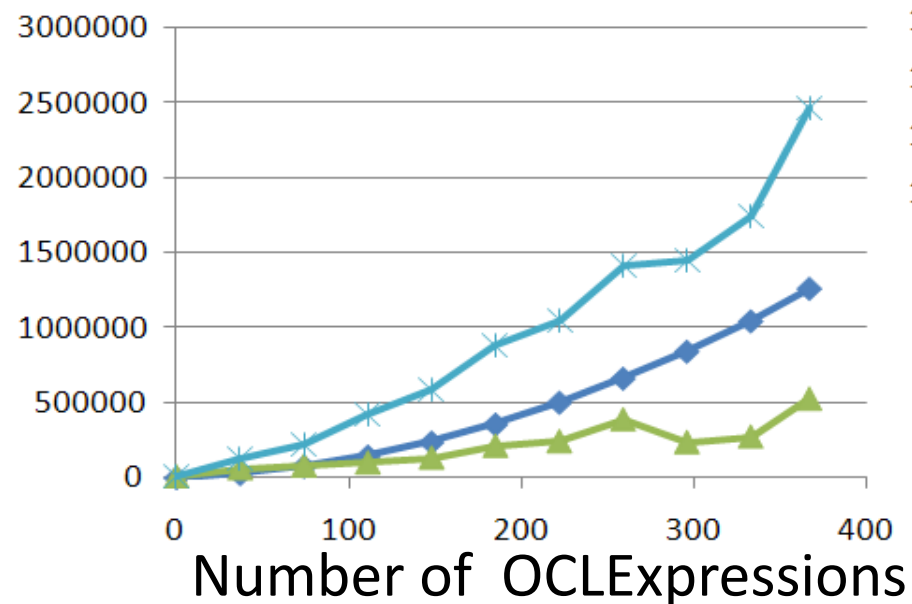
notifier.eClass() conforms to	Set<Adapter> interested
Parameter	[a1, a7, a15]
Signature	[a1, a3, a9]
...	...
⋮	⋮
feature	Set<Adapter> interested
NamedElement.name	[a3, a9, a14]
Call.signature	[a7, a15]
...	...

# Effects of HashMap-Based Eventing

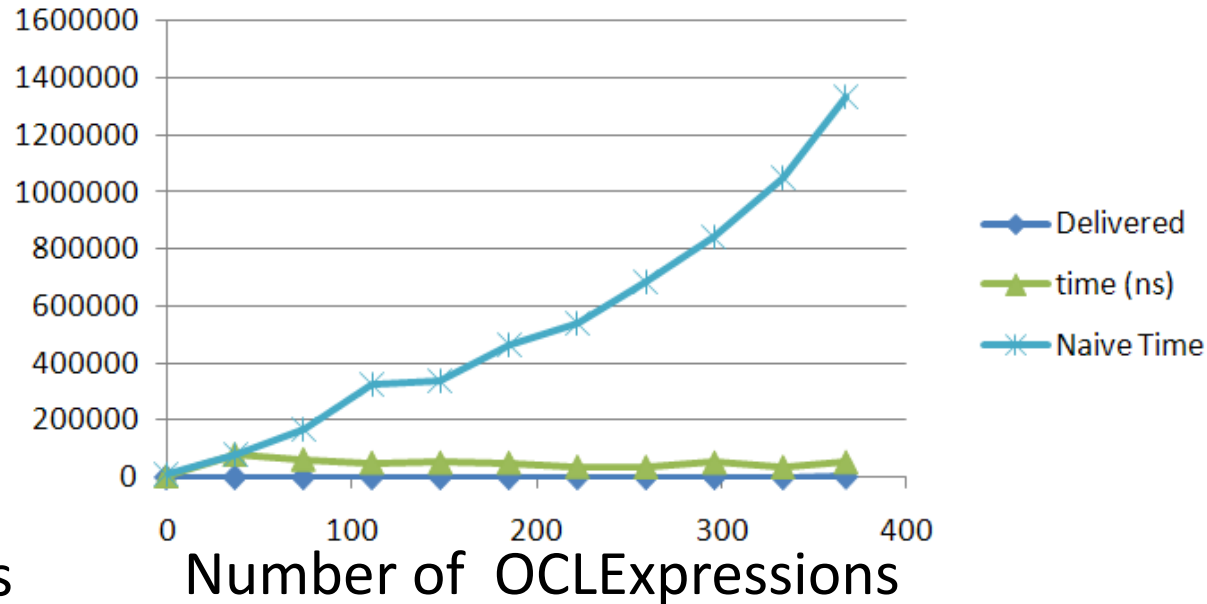
**Faster delivery** for Notifications matched by event filters

**No time increase** for expressions whose filters don't match a Notification

Notification #1

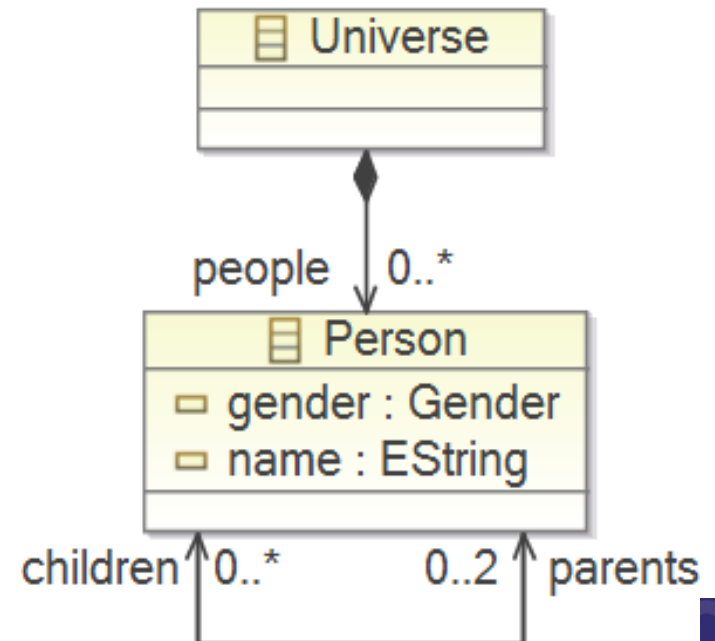


Notification #2

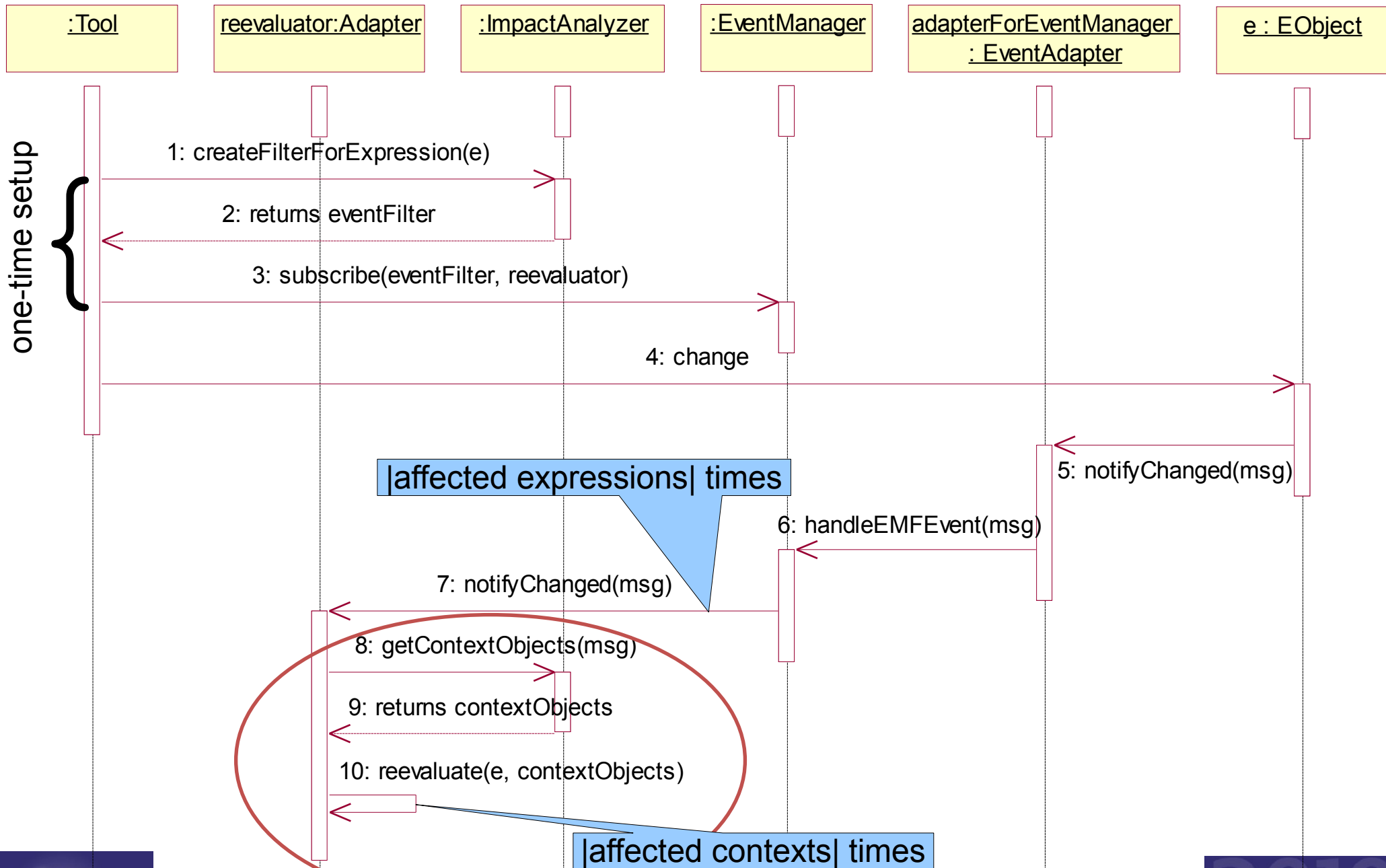


# Reducing Contexts for Re-Evaluation

- Use partial evaluation to prove value unchanged
  - `self.name='abc'` not affected by name change from 'x' to 'y'
- Use **Notification** object (`notifier`, `oldValue`, `newValue`) to navigate “backwards” to affected context objects
  - `self.children.children.name`
  - change attribute `name` on `x:Person`
  - contexts for re-evaluation:
    - `x.parents.parents`
- Tricky for iterators and recursive operations, but solved.



# Reduce Set of Context Elements

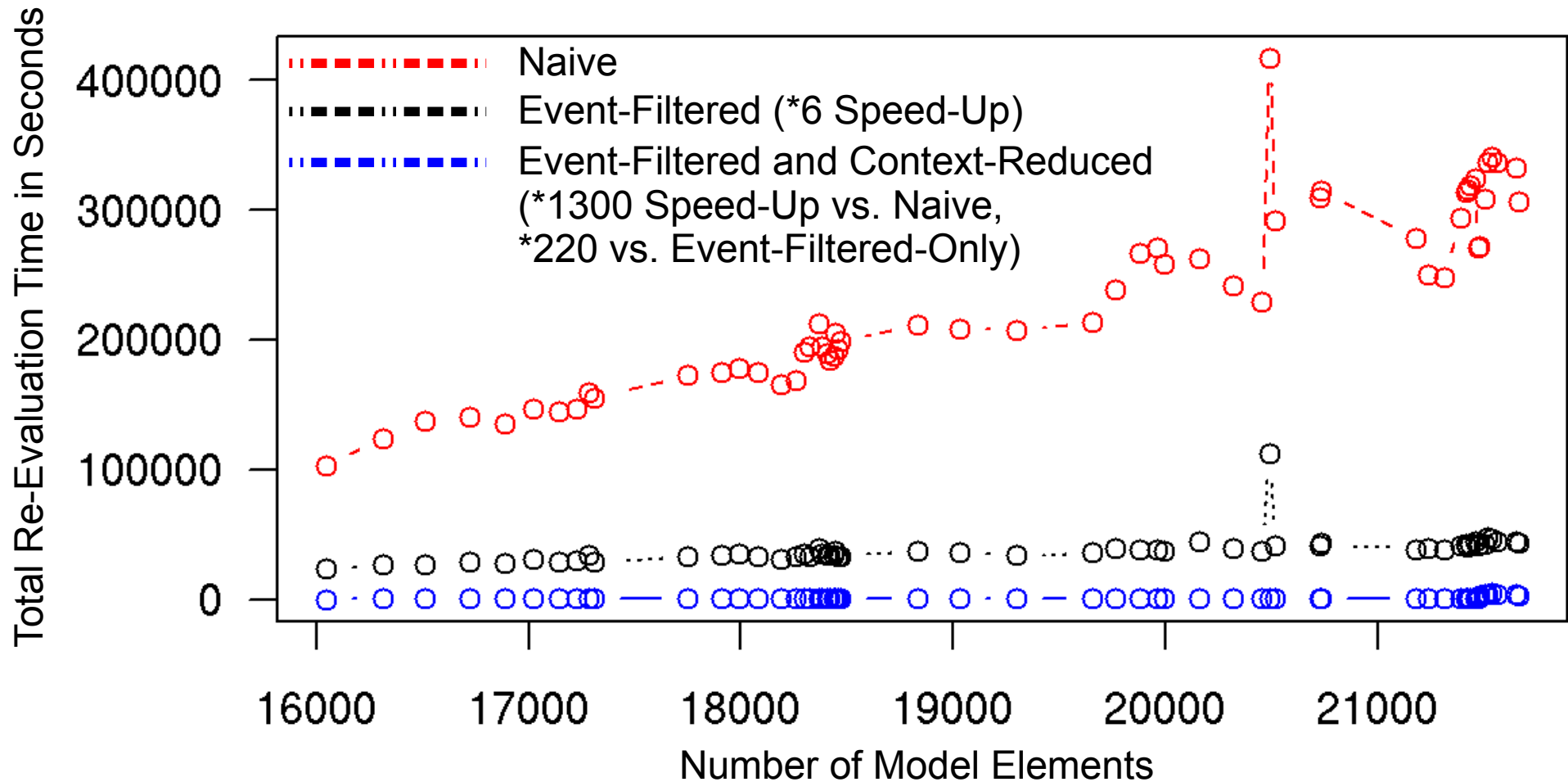




# API Usage Example

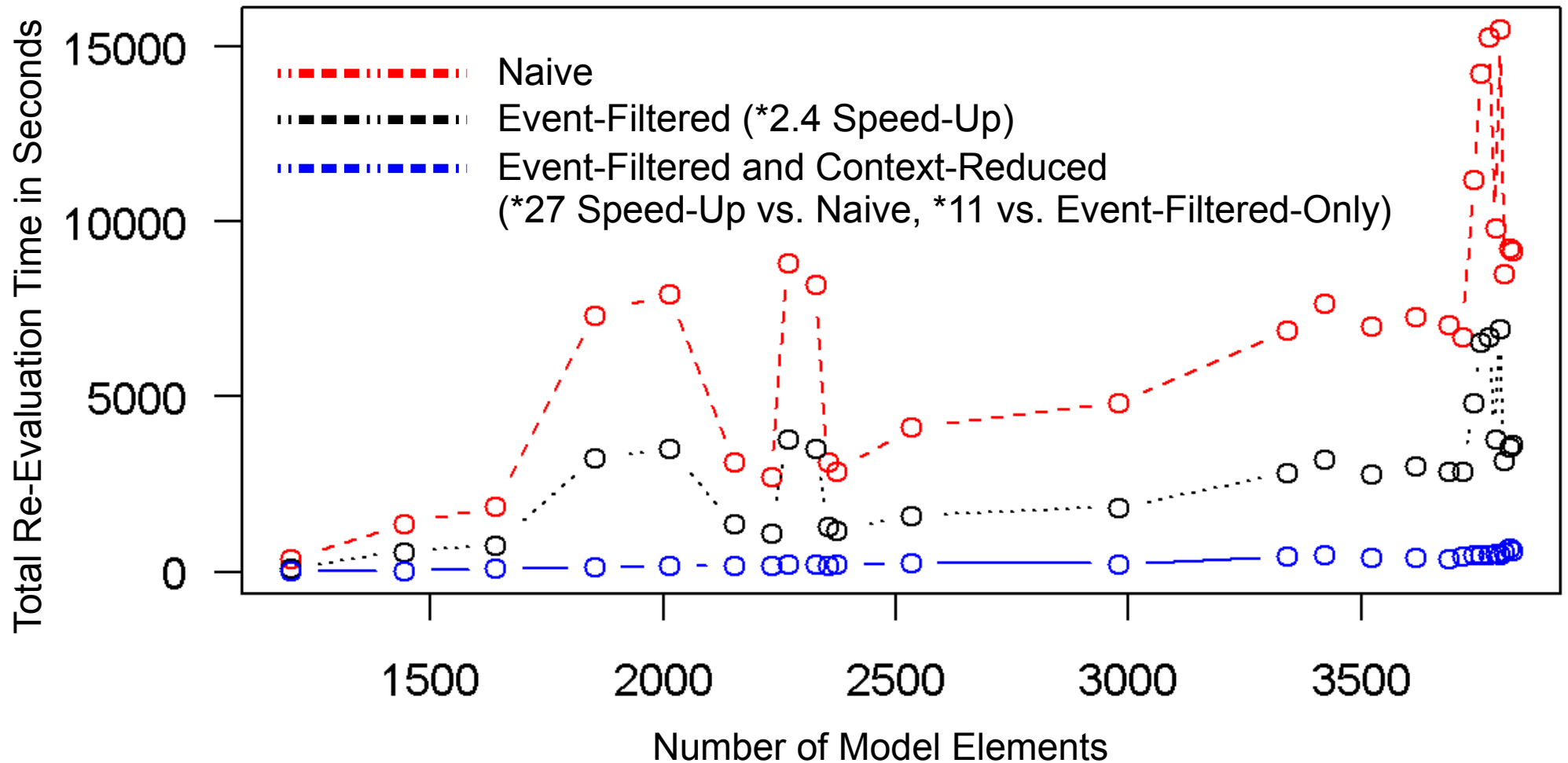
```
EventManager eventManager =
    EventManagerFactory.eINSTANCE.createEventManagerFor(
        editingDomain.getResourceSet());
final OCLExpression invariant = OCL.newInstance().createOCLHelper().
    createQuery("self.signature.parameters->size()=self.arguments->size()");
final ImpactAnalyzer impactAnalyzer =
    ImpactAnalyzerFactory.INSTANCE.createImpactAnalyzer(invariant,
        /* notifyOnNewContextElements */ true, oppositeEndFinder);
Adapter adapter = new AdapterImpl() {
    @Override
    public void notifyChanged(Notification msg) {
        // revalidate invariant on context objects delivered by impact analysis:
        Collection<EObject> revalidateOn = impactAnalyzer.getContextObjects(msg);
        if (revalidateOn != null && !revalidateOn.isEmpty()) {
            revalidate(invariant, revalidateOn);
        }
    }
};
eventManager.subscribe(impactAnalyzer.createFilterForExpression(), adapter);
```

# Benchmark Context Reduction (Average Case)



# Benchmark Context Reduction (Worst Case)

(apply changes to very central elements, referenced by all other model packages)



# Summary

MDT/OCL originally focussed on Java API  
Interactive Modeling Tools require OCL IDE

- EMF, Xtext, Acceleo, QVTo, OCL support richer OCL development environment

Extensibility required by QVTo, Acceleo

Efficiency required for serious use

IDE starting to appear

- Console, Editors

Expect/Demand much more

Contributions welcome

# OCL Resources

- OCL 2.2 Specification <http://www.omg.org/spec/OCL/2.2>
  - Clause 7 is quite readable (many typos)
- The Object Constraint Language:  
Getting Your Models Ready For MDA  
Jos B. Warmer, Anneke Kleppe
- Eclipse MDT/OCL project  
[http://www.eclipse.org/projects/project\\_summary.php?projectid=modeling.mdt.ocl](http://www.eclipse.org/projects/project_summary.php?projectid=modeling.mdt.ocl)
- Impact analysis
  - GIT: <http://anonymous@www.furcas.org/furcas.git/>
  - SVN: <https://www.hpi.uni-potsdam.de/giese/gforge/svn/bp2009>
  - Accounts: <https://www.hpi.uni-potsdam.de/giese/gforge/>