UML is still inconsistent!

How to improve OCL Constraints in the UML 2.3 Superstructure

Claas Wilke and Birgit Demuth

OCL 2011
Zurich, June 29th 2011
Motivation

- **OCL usage within the UML specification**
  - Definition of constraints
  - Definition of additional operations

- **When UML was originally specified, no OCL tooling existed**
  → OCL rules where specified manually (“by hand”)
  → No checks of syntax and static semantics

- **Investigations on UML/OCL inconsistency**
  - UML 2.0: 361 errors in 246 OCL rules [BGG04]
  - UML 1.5: 450 errors [FQL+03]
  - UML 1.3: 39 errors in 71 OCL rules [RG00]

- **This work**
  - Today’s situation (UML 2.3)?
  - How can similar problems avoided for future specifications?
Methodology (1/2)

Project online at http://www.dresden-ocl.org/index.php/DresdenOCL:WFRsInUML

Zurich, June 29th 2011
Methodology (2/2)

- Only OCL expressions from constraints sections investigated
  - Additional operations (body expressions) ignored
  - Context declarations were added manually

- Textual constraints were counted
  - Neither transformed into OCL
  - Nor checked if that is possible at all

- Erroneous constraints were fixed where possible

- Same error occurring multiple times in one constraint
  - Counted as one error
• 46.8% of all constraints have no OCL semantics!

• 48.5% of all OCL rules are erroneous!
Results
Different Types of Errors

- Total: 320 errors
- Classified into 14 types of five categories
Results

Syntactical Errors

1. Typing Errors
   15  (6.4 %)

2. Brackets
   27  (11.5 %)

3. Wrong Ifs
   8   (3.4 %)

4. Missing Escape
   14  (6.0 %)

5. Wrong use of #
   6   (2.6 %)

6. Wrong use of ->/.
   9   (3.8 %)

Zurich, June 29th 2011

UML is still inconsistent!
Results

Minor Inconsistencies

1. Wrong NamedElement Referred
   62 (26.4 %)

2. Operation vs. Property Call
   9  (3.8 %)

/* From UML 2.3 superstructure, clause 11.3.11.  
   (1) object.multiplicity.is(1,1) evolved to object.is(1,1). */
self.object.multiplicity.is(1,1)

/* From UML 2.3 superstructure, clause 15.3.8.  
   (2) PropertyCall instead of OperationCall: size. */
(self.kind = #initial) implies
(self.outgoing->size <= 1)
Results
Type Checking Errors

1. Result Type
   20  (8.5 %)

2. Wrong Iterator
   4   (1.7 %)

3. Missing asSet()
   15  (6.4 %)

4. Missing asOrderedSet()
   9   (3.8 %)

/* From UML 2.3 superstructure, clause 11.3.1.
   (1) Results in Bag(Boolean) since event is a collection
       (implicit collect()). */
trigger.event.oclIsKindOf(CallEvent)

/* From UML 2.3 superstructure, clause 7.3.4.
   (2) forEach must be used instead of collect. */
self.endType->excludes(self) and self.endType
   ->collect(et|et.allparents()->excludes(self))

/* From UML 2.3 superstructure, clause 7.3.4.
   (3) Implicit asSet() on ownedEnd does not work. intersection()
       expects a Set as its argument. */
ownedAttribute->intersection(ownedEnd)->isEmpty()
Results
Evolution Errors

1. Enumeration Literals
   54  (23.0 %)

2. Set{} vs. null
   16  (6.8 %)

/* From UML 2.3 superstructure, clause 7.3.15.
   (1) VisibilityKind::public/private must be used. */
   self.visibility = #public or self.visibility = #private

/* From UML 2.3 superstructure, clause 7.3.36
   (2) Null must be used instead of Set{}. */
   lower = if returnResult()->notEmpty() then returnResult()->any().lower else Set{} endif
Results
Implicit Conversions

1. Implicit asSet
   94

2. Implicit collect
   18

/* From UML 2.3 superstructure, clause 15.3.7. 
   (1) Implicit asSet() on effect. */
   effect->isEmpty()

/* From UML 2.3 superstructure, clause 11.3.23 
   (2) Implicit collect() on qualifier. */
   self.value->excludesAll(self.qualifier.value)
Lessons Learnt

- **The OCL rule quality has not been improved since [BBG04]!**

- **The current specification approach is insufficient**
  1. No syntactical checks
  2. No static semantics checks
  3. No dynamic semantic checks
  4. No Support for UML/OCL coevolution

- **Proposed specification improvements**
  1. Model-Based Specification process
  2. Elucidative Specification
  3. Use of OCL unit testing
  4. Use of UML/OCL coevolution tools

- **Proposed OCL improvements**
  1. Removal of -> operator
  2. Avoidance of implicit conversions
  3. Introduction of selectByKind()
Possible Improvements

Elucidative Specification

Zurich, June 29th 2011

UML is still inconsistent!
Possible Improvements
Removal of the -> operator

- The -> operator for collection operations is irritating
  - Even authors of UML do not know how to use it

- Using wrong operators is dangerous
  - Different semantics
  - Unnecessary conversions

```java
/* Two ways to invoke size() on Strings. */
name.size() > 1
name->size() > 1 -- means Set{name}->size()!

/* Two ways to invoke asSet() on objects. */
name.asSet()
name->asSet() -- means Set{name}->asSet()!
```

→ Use . for collection operations as well
Possible Improvements
Avoid implicit asSet and implicit collect

- **Implicit conversions often occur unforeseen**

```java
/* From UML 2.3 superstructure, clause 17.2.1. */

self.conveyed.represented->forAll(p | p->oclIsKindOf(Class)
  or oclIsKindOf(Interface) or oclIsKindOf(InformationItem)
  or oclIsKindOf(Signal) or oclIsKindOf(Component))

/* From UML 2.3 superstructure, clause 18.3.6. */

self.metamodelReference.importedPackage.elementImport
  .importedElement.allOwningPackages()
  ->union(self.metamodelReference.importedElement
    .allOwningPackages())->notEmpty()
```

- **Dangerous and unnecessary combinations possible**
  → They should be avoided, if not forbidden
Summary

- **Investigation of OCL rules defined in UML**
  - 53.2 constraints do not define any OCL rules
  - 48.5% of all OCL rules contain errors

- **Many errors could be avoided**
  - Improved specification process
  - Modifications of the OCL

- **Future Work**
  - Do other specifications have the same problems?
  - Are proposed improvements applicable?
  - How did existing UML implementations solved the OCL errors?
Literature (1/2)


Literature (2/2)


Thank you!

Dresden OCL
http://www.dresden-ocl.org/

Software Technology Group
http://st.inf.tu-dresden.de/

QualiTune
http://www.qualitune.org/

claas.wilke@tu-dresden.de
Backup
Results
OCL Rule Complexity (structure)

→ Most OCL rules are rather simple
Possible Improvements
OCL Unit Testing

- **Testing OCL constraints is sensible**
  - Do they constrain what they shall constrain?
  - Checks for runtime errors (invalid values)
    E.g., division by zero

- **OCL testing facilities exist**
  - [CO09], [HG10]

- **Unit tests could be deployed together with the specification**
  - Regression/acceptance tests for UML case tools
Possible Improvements
Introduction of selectByKind iterator

- A select is often followed by a cast
  - Filter $\rightarrow$ Cast

```c
// From UML 2.3 superstructure, clause 7.3.3. */
parents()->select(oclIsKindOf(Association))
   .oclAsType(Association)->forAll(p |
   p.memberEnd->size() = self.memberEnd->size())
```

- Requires two iterations $\rightarrow$ overhead

- Introduction of a selectByKind iterator

```c
// Modified from UML 2.3 superstructure, clause 7.3.3. */
parents()->selectByKind(Association)->forAll(p |
   p.memberEnd->size() = self.memberEnd->size())
```

$\rightarrow$ Only one iteration remaining
- And even more readable