Tool Supported OCL Refactoring Catalogue

Jan Reimann, Claas Wilke, Birgit Demuth, Michael Muck, Uwe Aßmann

OCL Workshop 2012, Innsbruck, 30/09/2012
Agenda

- Motivating example from finance sector
- Existing OCL refactorings and their limitations
- Extended catalogue and improvements
- Demo
- Conclusion and Future Work
Motivating Example

Bad evaluation performance

http://nomos-software.com/ Nomos XML validation service demo
Motivating Example

http://nomos-software.com/ Nomos XML validation service demo
Motivating Example

```ocl
class DirectDebitTransactionInformation1

context DirectDebitTransactionInformation1

inv EPC_AOS_DrctDbtTxInfDbtr:
let adr = dbtr.pstlAdr
in
  adr.adrTp = null and
  adr.strtNm = null and
  adr.bldgNb = null and
  adr.pstCd = null and
  adr.twnNm = null and
  adr.ctrySubDvsn = null and
  dbtr.ctryOfRes = null

endpackage
```

http://nomos-software.com/  Nomos XML validation service demo
OCL Refactoring

- **OCL-exclusive refactorings** and **OCL co-refactorings**

- Aim at improving qualities such as [Fowler 99]:
  - Reusability
  - Readability
  - Understandability
  - Comprehensibility
  - Maintainability
  - Evaluation performance

- For removing bad smells such as [Correa, Werner 07]:
  - Bad Evaluation performance
  - Deprecated null checks
  - Magic literal
  - Long journey

- One of the most demanded features in OCL IDEs [Chimiak-Opoka et al. 11]
Existing OCL-Exclusive Refactorings

- Automatic simplification of generated OCL constraints [Giese, Larsson 05]
  - Logical expressions are simplified
  - Example: \((\text{false} \text{ and exp}) \rightarrow \text{(false)}\)

- First OCL-exclusive refactoring catalogue [Correa, Werner 07]
  - Defined OCL smells...
  - ...upon which they specified according refactorings
Analysis Of Existing OCL-Exclusive Refactorings

- **Change Initial Navigation:**
  - Remove *Verbose Expressions or Long Journeys*
  - Lengthy navigation paths should be replaced by alternative shorter paths
  - Just propose to find alternative paths and pick the shortest one
  - → might not be semantics preserving!

[Correa, Werner 07], [Correa, Werner, Barros 09]
Analysis Of Existing OCL-Exclusive Refactorings

- Add Variable Definition/Replace Expression By Variable:
  - Two separate refactorings
  - Remove *Magic Literal*
  - A variable is initialised with a literal
  - Afterwards all occurrences are replaced by the variable
  - Not combined to one refactoring
  - Not mentioned that variable name uniqueness has to be verified
  - Not only literals but navigation paths can be replaced (like in the example)
  - We combined this refactoring to *Extract Variable*

[Correa, Werner 07], [Correa, Werner, Barros 09]
Extended Catalogue

- Corrected specifications of all refactorings from [Correa, Werner 07]

- Identified 4 categories:
  1. Renamings
  2. Removals/Materialisations
  3. Extractions/Inlinings
  4. Separations/Merges

- Identified further OCL-exclusive refactorings
# New OCL-Exclusive Refactorings

<table>
<thead>
<tr>
<th>Removals/Materialisation</th>
<th>Extractions/Inlinings</th>
<th>Separations/Merges</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remove deprecated null check</strong></td>
<td><strong>Extract/Inline variable</strong></td>
<td>Merge chained <code>let</code> expressions</td>
</tr>
<tr>
<td>Remove unused Elements</td>
<td>Extract Property/Operation</td>
<td>Split/Merge context declarations</td>
</tr>
<tr>
<td>Remove/Add redundant brackets</td>
<td>Inline Property/Operation</td>
<td></td>
</tr>
<tr>
<td>Remove/Materialise <code>self</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove/Materialise type declarations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove implicit <code>asSet</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove implicit <code>collect</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Catalogue can be seen here in the following days:

http://www.dresden-ocl.org/refactoring/
Demo
Conclusion And Future Work

Conclusion
• Analysed existing OCL-exclusive refactorings
• Identified limitations and improved them
• Specified a catalogue of 28 refactorings
• Implemented OCL-exclusive refactoring support for Dresden OCL [Wilke, Demuth 11]
• Used the generic refactoring framework Refactory [Reimann, Seifert, Aßmann 12]

Future Work
• Finish implementation
• OCL co-refactoring
The End

Thank You for Your Attention
Bibliography


[Wilke, Demuth 11] UML is still inconsistent! How to improve OCL Constraints in the UML 2.3 Superstructure. EASST, 2011
Contact

Jan Reimann
Technische Universität Dresden
Software Technology Group
jan.reimann@tu-dresden.de
http://st.inf.tu-dresden.de/

http://dresden-ocl.org/refactoring/

http://modelrefactoring.org/
Defining Generic Refactorings With Roles

Tool Supported OCL Refactoring Catalogue
Instantiation By Mapping The Roles

```
ROLEMODELMAPPING FOR <http://www.emftext.org/language/pl0>

"Extract Procedure" maps <ExtractXwithReferenceClass> {
    OrigContainer := Body {
        extracts := statements;
    };
    Extract := Statement;
    NewContainer := ProcedureDeclaration (newName -> name) {
        moved := block -> body -> statements;
    };
    MovedReference := CallStatement {
        containerRef := procedure;
    };
    ContainerContainer := Block {
        source := body;
        target := procedures;
    };
}
```
Mapping To Paths

Classifier

SuperElement

SubElement

SuperElement

Generalization

SubElement SuperElement

1 specific

generalization 0..*

1 general