Design and Prototypical Implementation of a Pivot Model as Exchange Format for Models and Metamodels in a QVT/OCL Development Environment

Großer Beleg – Final Presentation
Contents

- Introduction
- Research Methodology
- Results
- Evaluation
Contents

- Introduction
- Research Methodology
- Results
- Evaluation
Motivation

- Model-driven Software Development (MDSD)
  - emerging paradigm for higher productivity and quality in software engineering
- increasing importance of domain-specific modeling languages (DSL)
  - on meta layers M2 and M3
- requires precise models, model transformations
- idea: use a standard constraint and model query language like OCL on instances of arbitrary DSLs
Goals

- design of a pivotal metamodel to integrate OCL with multiple DSLs

- implement an approach to realize the mapping between the Pivot Model and the target DSL
Example

- **EMF Ecore**
  - small and specialized language for defining object-oriented metamodels
  - meta-metalanguage (M3)

- **Benefits of integration with OCL:**
  - express wellformedness rules over Ecore models
  - transform Ecore models using QVT

- **Example language defined with Ecore:**
  - Plugin Modeling Language (PML)
- Example language defined with Ecore:
  - Plugin Modeling Language (PML)
Contents

- Introduction
- Research Methodology
- Results
- Evaluation
Approach

- analysis of literature about metamodeling to identify foundational challenges
- analysis of related work to identify respective strengths and weaknesses:
  - Dresden OCL2 Toolkit
  - Kent OCL Library
  - Epsilon Platform
- definition of a conceptual framework to guide research
Conceptual Framework

- Concepts Level
- Definition Level
- Execution Level
Applying the Framework

- requires integration of Ecore with OCL
Contents

- Introduction
- Research Methodology
- Results
- Evaluation
Design of the Pivot Model
Implementation of Model Adaptation

- OCL engine knows
- Pivot Model operates on
- Target DSL «realize»
- Adapters adapt
Presentation

- generically displaying Ecore models through the Pivot Model interface
Integrating the OCL Standard Library

- **OCL Standard Library**
  - predefined types and operations

- **some problems with integration:**
  - infinite number of OCL collection and tuple types
  - all model types implicitly derive from `OclAny`

- **existing OCL engines:**
  - dynamic creation of Standard Library in the code
  - complex, error-prone
Integrating the OCL Standard Library

- my solution:
  - support templates (generics) in the Pivot Model
  - model Standard Library as instance of Pivot Model
  - integrate by loading serialized XMI and bind generic types when necessary
Presentation

- modeling the OCL Standard Library
Writing OCL expressions for Ecore models

- example: a wellformedness rule for PML

```
-- a Plugin must have a valid id
context Plugin
inv: self.id->notEmpty()
```

- problem:
  - existing OCL parser needs adaptation

- solution:
  - alternative concrete syntax for OCL based on XML
  - use EMF for serialization / deserialization
OCL in XML

- XOCL ... XML-based OCL

```xml
<xocl:NamespaceXS pathName="pml">
  <ownedRule name="idNotEmpty" kind="invariant" constrainedElement="Plugin">
    <specification body="self.id->notEmpty()">
      <bodyExpression xsi:type="xocl:CollectionOperationCallExpXS"
        referredCollectionOperation="notEmpty">
        <source xsi:type="xocl:PropertyCallExpXS" referredPropertyName="id">
          <source xsi:type="xocl:VariableExpXS"
            referredVariable="//@ownedRule.0/@specification/@context"/>
        </source>
      </bodyExpression>
    </specification>
  </ownedRule>
</xocl:NamespaceXS>
```
Presentation

- visually creating OCL expressions

We will create a new OCL expression in XOCL syntax.
Parsing OCL expressions

- adapter layer allows to add *transient* elements to a domain-specific model
  - Constraint instances representing OCL expressions
  - properties and operations defined by OCL expressions
Presentation

- parsing an XOCL file

Finally, we want to parse OCL constraints into our adapted model. The PML metamodel has already been loaded into the Model Browser view.
Contents

- Introduction
- Research Methodology
- Results
- Evaluation
### Evaluation

- comparison of effort to integrate a DSL

<table>
<thead>
<tr>
<th></th>
<th>Dresden OCL2 Toolkit</th>
<th>Kent OCL</th>
<th>Pivot Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapted metamodel</td>
<td>UML</td>
<td>MOF</td>
<td>Ecore</td>
</tr>
<tr>
<td>Lines of code</td>
<td>2124</td>
<td>1657</td>
<td>685</td>
</tr>
</tbody>
</table>

- automatic *generation* of large parts of Pivot Model adapter layer possible
Contributions

- detailed analysis of conceptual challenges
- proposal of a conceptual framework
- thorough review of current Dresden OCL2 Toolkit
- carefully designed Pivot Model
- novel approach for integrating Standard Library
- clean and highly extensible design of an integration framework
- investigation of Execution Level in preparation of future developments (OCL interpreter)
The End

Thank you for your attention! 😊
Questions? Comments?
Backup
Ontological Classification Problem

- two dimensions of metamodeling
Ontological Classification Problem

- In UML: Stereotypes and Profiles extend M2 concepts
- DSLs define entirely new ontology concepts on M2
System Instantiation Problem

- Transformation on the System layer requires instantiation of new System elements
- Instantiation semantics?

Model Space

System Space
Repository Adaptation in Dresden OCL Toolkit

- UML-specific
- monolithic
  (33 methods)

ModelFacade

- `getRefObject (mofID : String) : Object`
- `getFeature (mofID : String) : List`
- `getName (mofID : String) : String`
- `getMultiplicity (mofID : String) : Multiplicity`
- `getOrdering (mofID : String) : OrderingKind`
- `getQualifier (mofID : String) : List`
- `getNamespace (mofID : String) : Namespace`
- `getUpper (mofID : String) : int`
- ...

HashMap<String, Object> `refObjects`

<table>
<thead>
<tr>
<th>mofId</th>
<th>refObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>7D749D32-...:00036B</td>
<td></td>
</tr>
<tr>
<td>7D749D32-...:000364</td>
<td></td>
</tr>
<tr>
<td>7D749D32-...:00035E</td>
<td></td>
</tr>
<tr>
<td>7D749D32-...:000352</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Netbeans Metadata Repository (MDR)

Custom Repository
OclFactory

- `getOclRepresentationFor(type : OclType, o : Object) : OclRoot`
- `reconvert(targetType : NonOclType, oclObject : OclRoot) : Object`
- `getOclModelTypeFor(pathname : String) : OclModelType`
- `getOclEnumTypeFor(pathname : String) : OclEnumType`
- `getOclTupleType(names : String[], types : OclType[]) : OclTupleType`

OCL Space

- OclModelObject
- OclEnumLiteral
- OclModelType
- OclSequence

Java Space

- java.lang.Object
- java.lang.Class
- java.lang.Enum
- java.util.List

Mapping