Development of an OCL-Parser for UML-Extensions

Closure of a Diploma Thesis
Fadi Chabarek
Introduction

• Short explanation of the subject and its main technologies

• Introduction to the developed solution:
  – framework’s architecture
  – model interface
  – parser, context checker, interpreter
  – MOF Bridge
OCL

• Semiformal Constraint Language
• Part of the UML
• Supports invariants, pre and post conditions
• Constraints are defined for types / model elements:

```plaintext
context Company inv:
    self.numberOfEmployees > 50

context Person::income(d : Util::Date) : Integer
    post: result = 5000
```
Metamodeling

- M3 layer: meta-metamodel
  - MOF Model

- M2 layer: metamodels
  - UML Metamodel
  - IDL Metamodel

- M1 layer: models
  - UML Models
  - IDL Interface

- M0 layer
Connection: UML Profiles

• Extension system of UML
• Define additional constraints to the UML Metamodel
• Narrow models down to domain specific requirements
• Constraints described through OCL can be validated
Diploma Subject

• OCL expressions have to be interpreted in the context of UML Profiles and the UML-Metamodels 1.3 and 1.4.

• Therefore a „Parser“ has to be developed, which:
  – gets an UML model instance and an UML-Profile as input
  – and validates the adherence of the model to OCL constraints defined by the given UML-Profile
Architecture

Diagram showing the architecture of a system involving:
- OCL File
- XMI
- MOF Repository
- Model Abstraction Layer
  - Parser
  - Context Checker
  - Interpreter
- Report

Validator
Model Interface

• Abstraction Layer to interpret OCL constraints in context of arbitrary models
• Designated to be implemented for MOF compliant metamodels
• Enables support of different versions of the UML Metamodel
The Interface‘s Basic Idea

• OCL is defined in the context of UML
• OCL augments its type system through model types via UML concepts (e.g. UML Classifier, Properties etc.).
• Description of these concepts in a model define the model‘s OCL semantics
Structure of the Model Interface

Facade describes the model on its:

- Type level:
  - Packages
  - Classifiers
  - Properties

- Instance level:
  - Instances
  - Reflective Properties for OCL meta level Operations
Parser

- OCL Grammar does not produce a LR-Language
- Changes to the grammar are necessary
- Choice of parser generator: SableCC
- Enlargement of the language is circumvented by concrete syntax
- New grammar is LALR(1), the parser accepts the same language
Context Checker

• OCL type system consists of predefined and model types
• Java Interfaces describe predefined types.
  – Instances implement these interfaces
  – Java Reflection API resolve the interface’s properties.
  – This allows later changes to the OCL type system to be reflected
• Model types and their properties are resolved through the model interface
Visitor Pattern

- SableCC generates Parser and Visitors.
- When a Visitor visits a node in three phases:
  - in ... is called when entering a node
  - case ... lets the visitor visit the node's children
  - out ... is called when leaving a node
Type check

• Implementation of a static type check
• Usage of the Visitor-Pattern
• AST is traversed bottom-up from left to right by overriding out methods.
• Exceptions of this order are implemented by redefining case methods.
• Types are determined at the bottom of the tree and used in the parent nodes until the root is reached
Interpreter

- Corresponding to the type system there are predefined and model instances
- Predefined instances are implemented on the basis of the type interfaces
- Model instances delegate to the model interface
Evaluation

- Values are evaluated bottom-up from left to right.
- Constraints must be evaluated for every instance of a type.
- The respective constraint holds if the root node evaluated to true.
MOF Bridge

- Java Metadata Interface (JMI)
  - MOF Mapping for Java
  - MDR implementation supports import of metamodel over XMI
- JMI enables access to MOF compliant metamodels
- MOF Bridge connects the model interface with JMI
Sequence

validator

mofFacade (MOF Bridge)

mofMetadata (JMI)

request (OCL/UML concept)

response (OCL/UML concept)

request (MOF concept)

response (MOF concept)
What did we actually achieve?

- Concrete JMI technologies (e.g. MDR) represent the UML Metamodels 1.3 and 1.4 and its instances
- This representation is translated by the Abstraction Layer of the framework:
  - MOF to UML by the MOF Bridge
  - UML to OCL by the model interface and the framework
- OCL semantics are stipulated for the UML Metamodels.
- Constraints can now be validated by the framework
By-Products

The Abstraction Layer of the framework facilitates:

• Support of OCL for arbitrary models
• Support of OCL for MOF compliant metamodels
• The definition of a general OCL tool interface
Conclusion: Summary 1

• Presentation of UML Profiles and the subject of the diploma thesis

• Model interface
  – Basic Idea
  – Type and instance level

• Parser
  – Changes to the grammar, LALR(1)
Conclusion: Summary 2

• Context Checker
  – Implementation of the type level
  – Description of the static type check algorithm

• Interpreter
  – Implementation of the instance level
  – Description of the evaluation algorithm

• MOF Bridge
  – JMI