RoclET– A Tool for Wrestling with OCL Specifications

Cédric Jeanneret and Leander Eyer and Slaviša Marković and Thomas Baar

École Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland
{cedric.jeanneret, leander.eyer, slavisa.markovic, thomas.baar}@epfl.ch

Abstract. In this paper, we describe the architecture and the functionality of our own OCL tool called RoclET[1]. Besides standard features of OCL tools such as editing of class and object diagrams and parsing of OCL assertions (invariants, pre-/post-conditions), our tool supports also the evaluation of OCL constraints in a given system snapshot (object diagram), refactoring of UML/OCL models, and impact analysis. RoclET is deployed in form of an Eclipse plugin.

1 Precise Modeling with OCL

The Unified Modeling Language (UML) is today the most popular object-oriented modeling language for software systems. UML is in the first place a graphical notation what makes software models easily accessible by humans. UML diagrams can give a good overview on the modeled software system, but there is a lack of expressive power once the details of the software system have to be captured as well. A prevailing practice to resolve this problem is to add comments to UML diagrams and to clarify the intended meaning using natural language. Such informal comments, however, do not alter the formal meaning of the model and are ignored by tools when processing the model, e.g. in order to generate code. Another disadvantage is, that reading informal comments can become easily a hard and also ambiguous task once the comments are a little bit more complex.

The Object Constraint Language (OCL), see [1] for both an introduction and the language specification, is a textual language with formal syntax and semantics. OCL constraints capture a wide range of details that software developers wish to express in precise software models. The main application scenario are UML class diagrams. Here, OCL constraints can express conditions that should be obeyed in each system state (invariants) and contracts for system operations (pre-/post-conditions).

Most of the current OCL tools – USE [2], Octopus[3], Dresden OCL Toolkit [4] and OCLE [5] being the most influential ones – were developed in academia. Whereas almost each tool offers, besides parsing facilities for OCL, a functionality to generate implementation stubs out of UML/OCL models, relatively little

[1] RoclET is freely available from www.roclet.org

effort has been made so far to analyze the OCL constraints themselves, to provide functionalities for automatic constraint simplification, for refactoring, for analyzing of which impact a (small) change in a snapshot on the validity of a given OCL invariant has.

RoclET aims at providing facilities for a painless authoring, processing and analysis of OCL constraints. The main functionalities of RoclET are[2]:

- Parsing and type analysis[3]
- Evaluation in a given object diagram (applying the technique described in [6])
- Refactoring of UML class diagrams including necessary changes on attached OCL constraints (see [7])
- Blocking state analysis for operation contracts
- Impact analysis of changes made in the object diagram wrt. the validity of OCL constraints (this feature is based on the pioneering work of Cabot and Teniente [8, 9])

2 Architecture of RoclET

We have chosen a 3-layer architecture for RoclET (comp. Fig. 1): presentation layer, application layer and data layer.

The *presentation layer* consists of the editors and views the user interacts with. Due to a lack of flexible editors for class and object diagrams, we have decided to implement our own editors. The presentation layer has direct access to the *data layer* where the models are stored in a repository as instances of the UML/OCL metamodel. Currently, the versions UML 1.5 and OCL 2.0 are supported.

RoclET's functionalities are implemented in the *application layer*, mainly in form of transformation rules written in QVT. These transformations work on the repository and usually alter it directly.

2.1 Technologies

The architecture shown in Fig. 1 is a birds eye view on the system and does not reveal much of the used technology. The layered approach makes the architecture stable even if in future releases of our tool we would decide to realize parts of the tools on a different technology.

Before OCL constraints can be processed by our tool, they have to be transformed from their textual notation into an instance of the OCL metamodel. For this task we rely on the Dresden OCL parser [4]. Unfortunately, this parser can currently not handle the most recent UML version 2.0. Another restriction is that only a MOF-based MDR repository is supported for storing the parsed OCL constraints. The latter restriction was a serious problem we had to solve when implementing RoclET as a plugin for Eclipse, which is designed to work with Ecore-based repositories.

[2] Some of the functionalities are not fully implemented yet.
[3] Currently, the Dresden-OCL parser [4] is integrated for this purpose.

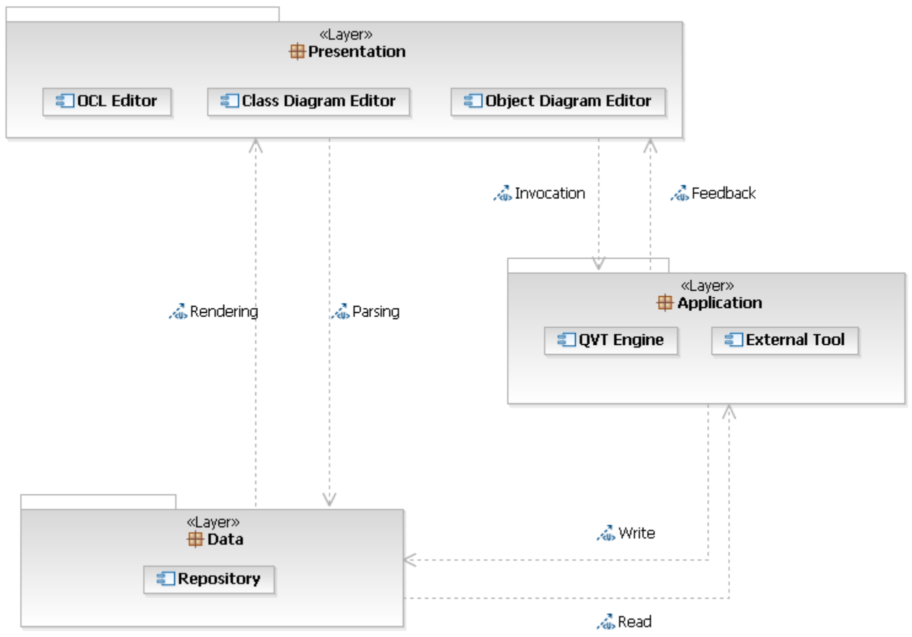

Fig. 1. Architecture

References

1. OMG. UML 2.0 OCL Specification – OMG Final Adopted Specification. OMG Document ptc/03-10-14, Oct 2003.
2. USE team. USE homepage. http://www.db.informatik.uni-bremen.de/projects/USE/, 2006.
3. OCTOPUS team. OCTOPUS homepage. http://octopus.sourceforge.net/, 2006.
4. Dresden-OCL team. Dresden-OCL homepage. http://dresden-ocl.sourceforge.net/, 2006.
5. OCLE team. OCLE homepage. http://lci.cs.ubbcluj.ro/ocle/, 2006.
6. Slaviša Marković and Thomas Baar. An OCL semantics specified with QVT. In Oscar Nierstrasz, Jon Whittle, David Harel, and Gianna Reggio, editors, *Proceedings, MoDELS/UML 2006, Genova, Italy, October 1-6, 2006*, volume 4199 of *LNCS*, pages 660–674. Springer, October 2006.
7. Slaviša Marković and Thomas Baar. Refactoring OCL annotated UML class diagrams. In *MoDELS'05*, volume 3713 of *LNCS*, pages 280–294. Springer, 2005.
8. Jordi Cabot and Ernest Teniente. Computing the relevant instances that may violate an OCL constraint. In Oscar Pastor and João Falcão e Cunha, editors, *17th International Conference on Advanced Information Systems Engineering, CAiSE 2005, Porto, Portugal, June 13-17, 2005*, volume 3520 of *LNCS*, pages 48–62. Springer, 2005.
9. Jordi Cabot and Ernest Teniente. Incremental evaluation of OCL constraints. In *18th International Conference on Advanced Information Systems Engineering,*

CAiSE 2006, Luxembourg, June 5-9, 2006, volume 4001 of *LNCS*, pages 81–95. Springer, 2006.