

Structure of the Dresden OCL Toolkit

Extended Abstract

Birgit Demuth
Dresden University of
Technology
Department of Computer
Science
Dresden, Germany
Birgit.Demuth@inf.tu-
dresden.de

Sten Loecher
Dresden University of
Technology
Department of Computer
Science
Dresden, Germany
Sten.Loecher@inf.tu-
dresden.de

Steffen Zschaler
Dresden University of
Technology
Department of Computer
Science
Dresden, Germany
Steffen.Zschaler@inf.tu-
dresden.de

The Object Constraint Language (OCL) as a part of the UML standard [1] is a formal language for defining constraints on UML models. We present a software platform for OCL tool support [2]. The platform is designed for openness and modularity, and is provided as open source. The goal of this platform is, for one thing, to enable practical experiments with various variants of OCL tool support, and then, to allow UML tool builders and users to integrate and adapt the existing OCL tools into their own environments. The Dresden OCL Toolkit provides the following tools:

OCLCore: The base tool of the OCL toolkit consists of four different modules:

- The **OCLParser** transforms the input OCL expression into an abstract syntax tree [3]. The abstract syntax tree forms the common data representation for all other tools in the toolkit.
- The **OCLEditor** is a comfortable editor which includes, besides editing of constraints, features like a toolbar and adequate error messages. The user interface is designed to allow the integration of the OCL editor not into a specific UML tool, but into various environments. The screenshot in Figure 1 gives an impression of the OCL editor integrated into Together. It shows on the right hand side a UML class diagram for a simple hotel reservation system. On the left hand side, an OCL constraint has been added asserting that the region of a hotel must be the same as the region of the hotel's destination.
- The **OCLTypeChecker** checks type correctness of OCL expressions and offers type information towards other modules. Necessary UML model

information has to be extracted from the environment in which the OCL toolkit is embedded. For this purpose a small external interface (called **ModelFacade**) is provided [3].

- The **OCLNormaliser** transforms the abstract syntax tree into a *normal form* of OCL terms, such that all terms can be mapped into a subset of the OCL language more adequate for subsequent tasks. That way it can be avoided that every tool using OCLCore has to implement the execution of every OCL expression completely.

OCL2Java: This tool transforms a normalised syntax tree into Java Code. It uses a class library which offers Java representations for the predefined OCL types.

OCLInjector4Java: The tool takes the generated Java code and inserts it into an application program. This code instrumentation is done by the generation of wrapper methods for all methods whose compliance to specified OCL constraints is to be checked during execution. The used technique including code cleaning is described in [6]. OCLInjector4Java has been integrated into ArgoUML and Together.

OCL2SQL: The SQL code generator [4] generates an SQL check constraint, assertion or trigger for an OCL invariant based on the accordingly normalised abstract syntax tree. OCL2SQL can be used and adapted for different relational database systems and different object-to-table mappings. Similarly to OCLTypeChecker's ModelFacade, we provide an interface for the integration of various strategies of object-to-table mapping.

OCLInterpreter: A first tool developed outside of the Dresden University of Technology is an OCL interpreter that allows the dynamic checking of OCL constraints against objects. The OCLInterpreter is also designed based on a normalised abstract syntax tree.

OCL20: All previous tools establish an architecture which is designed for OCL 1.x support. Currently we are reengineering the Dresden OCL Toolkit according to the new requirements of the revised and approved specification of OCL ("OCL 2.0" [7]). The OCL20 module

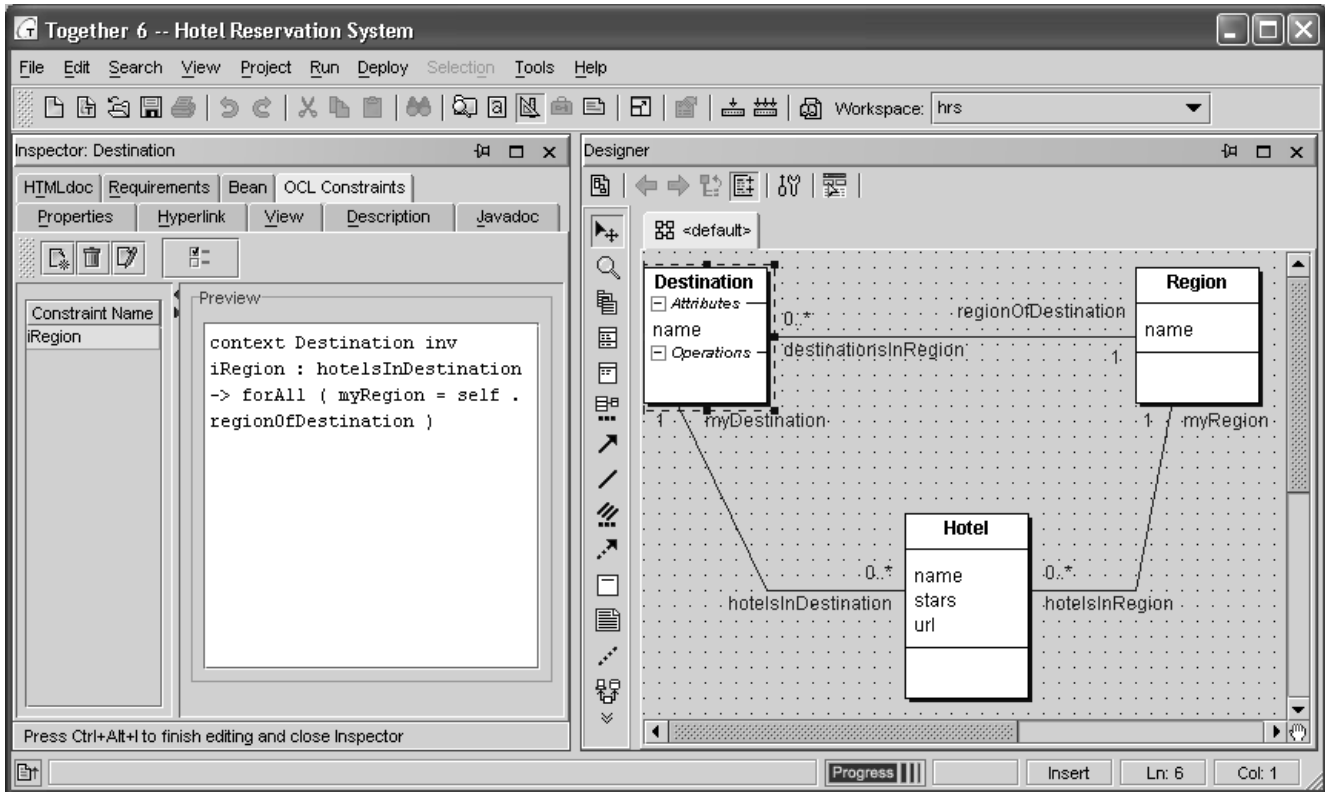


Figure 1: Dresden OCL Toolkit integrated into Together

is a prototype of a metamodel-based OCL compiler consisting of a MOF repository implementation and a code generator [5]. The OCL 2.0 parser is still under development. The research issue is to which extent a parser can be automatically generated from the provided specification.

An important requirement on tools supporting OCL is their cooperation with UML tools. The specification of OCL constraints without any model makes no sense. OCLType-Checker's ModelFacade provides support for this flexibility. We have implemented the ModelFacade in the following ways: An OCL tool can be **tightly** integrated into a UML tool as an add-in. Then the model interface must be implemented by an integration component accessing the UML tool's repository. Examples for this technique are the integration of our toolkit into Together (see Figure 1), ArgoUML, Poseidon, and Rational Rose. A kind of **loose** integration is the use of XMI files for static UML model information. The Dresden OCL toolkit already provides the necessary ModelFacade implementation to use this technology.

1. REFERENCES

- [1] OMG UML v. 1.5 specification, www.omg.org/technology/documents/formal/uml.htm
- [2] Dresden OCL Toolkit, <http://dresden-ocl.sourceforge.net/>
- [3] Hussmann, H., Demuth, B., Finger, F.: Modular Architecture for a Toolset Supporting OCL. in: Third

Int. Conference on the Unified Modeling Language (UML'2000), York, UK, October 2000, Springer, 2000

- [4] Demuth, B., Hussmann, H., Loecher, St.: OCL as a Specification Language for Business Rules in Database Applications. in: Fourth Int. Conference on the Unified Modeling Language (UML 2001), Toronto, Canada, October 1-5, 2001
- [5] Loecher, St., Ocke, St.: A Metamodel-Based OCL-Compiler for UML and MOF. in: Workshop OCL 2.0 - Industry standard or scientific playground?, Sixth Int. Conference on the Unified Modelling Language (UML 2003), October 21, 2003, San Francisco, www.ilkd.uni-karlsruhe.de/~baar/oclworkshopUml03
- [6] Wiebicke, R., Utility Support for Checking OCL Business Rules in Java Programs. Masters Thesis, Dresden University of Technology, 2000, dresden-ocl.sourceforge.net/
- [7] OCL 2.0 Submission, www.klasse.nl/ocl/ocl-subm.html