8th International Workshop on OCL Concepts and Tools

Jordi Cabot[1], Martin Gogolla[2], and Pieter Van Gorp[3]

[1] Open University of Catalonia (Spain), jcabot@uoc.edu
[2] University of Bremen (Germany), gogolla@informatik.uni-bremen.de
[3] University of Antwerp (Belgium), pieter@pietervangorp.com

Abstract. This paper reports on the 8th OCL workshop held at the MODELS conference in 2008. The workshop focussed on how to evaluate, compare and select the right OCL tools for a given purpose and how to deal with the expressiveness and complexity of OCL. The workshop included sessions with paper presentations as well as a special tool demo session.

1 Introduction

In recent years, MDA and associated MDE methodologies, approaches and languages (like QVT) emphasized the role that the Object Constraint Language (OCL) has to play in MDE development. Moreover, the modeling community is continuously pushing forward the OCL, far beyond its initial requirements as a precise modeling language complementing UML descriptions. Now, OCL is used in quite different applications domains (e.g., domain-specific languages and web semantics) and for various purposes (e.g., model verification and validation, code-generation, test-driven development, and transformations). To be successful, all these new OCL applications, extensions and usages require new OCL tools that support them.

This workshop aimed to look specifically at how to develop, apply, evaluate and compare all kinds of OCL-related tools. The workshop brought together OCL practitioners and OCL tool builders (from both academy and industry) in order to evaluate today's state-of-the-practice. In particular, the workshop focused the discussion on how to evaluate, compare and select the right OCL tools for a given purpose, how to deal with the expressiveness and complexity of the language and how to tackle its ambiguous or underdefined issues from a practical point of view. The workshop discussed new OCL tools and patterns, libraries, and algorithms that may facilitate development and reuse. In addition, all other aspects that may improve the adoption and support of OCL or its usability were considered. As a result, the workshop contributed to consolidate and expand the role of OCL in the modeling community by discussing approaches for research and development that could potentially drive the building of new OCL tools.

All submitted papers were reviewed by three industrial or academic members from the Program Committee:

- David H. Akehurst, Thales, UK
- Thomas Baar, Tech@Spree, Germany
- Jean Bézivin, University of Nantes, France
- Behzad Bordbar, University of Manchester, UK
- Achim Brucker, SAP, Germany
- Dan Chiorean, University of Cluj, Romania
- Tony Clark, Ceteva, UK
- Birgit Demuth, Technical University of Dresden, Germany
- Remco Dijkman, Eindhoven University of Technology, The Netherlands
- Robert France, University of Fort Collins, USA
- Heinrich Hussmann, University of Munich, Germany
- Richard Mitchell, Inferdata, UK
- Mark Richters, Astrium Space Transportation, Germany
- Shane Sendall, IBM, Switzerland
- Burkhart Wolff, University of Paris-Sud (Orsay), France
- Steffen Zschaler, Lancaster University, UK

2 Workshop Papers

Here we give a short overview of the papers presented in the workshop taken from the abstracts provided by the authors. The papers were organized in three different sessions: *Executing OCL* (papers 1 to 3), *Translating OCL* (4 to 6) and *Validation and Verification* (7 to 9). Polished versions of the workshop papers will appear in an EASST special issue.

2.1 Building an efficient component for OCL evaluation by Manuel Clavel, Marina Egea and Miguel García de Dios

In this paper we report on our experience implementing the Eye OCL Software (EOS) evaluator, a Java component for efficient OCL evaluation. We first motivate the need for an efficient implementation of OCL in order to cope with novel usages of the language. We then discuss the performance of the EOS component when evaluating expressions on medium-large scenarios. Finally, we explore various approaches for evaluating OCL expressions on really large scenarios.

2.2 Static Source Code Analysis using OCL by Mirko Seifert and Roland Samlaus

The majority of artifacts created during software development are representations of programs in textual syntax. Although graphical descriptions are becoming more widespread, source code is still indispensable. To obtain programs that behave correctly and adhere to given coding conventions, source code must be analyzed — preferably using automated tools. Building source code analyzers has a long tradition and various mature tools exist to check code written in conventional languages, such as Java or C. As new languages emerge (e.g., Domain

Specific Languages) these tools can not be applied and building a tool for each language does not seem feasible either. This paper investigates how meta models for textual languages and the Object Constraint Language can enable generic static source code analysis for arbitrary languages. The presented approach is evaluated using three languages (Java, SQL and a DSL for state machines).

2.3 Optimization patterns for OCL-based model transformations by Jesús Sánchez Cuadrado, Frédéric Jouault, Jesús García-Molina and Jean Bézivin

Writing queries and navigation expressions in OCL is an important part of the task of developing a model transformation definition. When such queries are complex and the size of the models is significant, performance issues cannot be neglected. In this paper we present five patterns intended to optimize the performance of model transformations when OCL queries are involved. For each pattern we will give an example as well as several implementation alternatives. Experimental data gathered by running benchmarks is also shown to compare the alternatives.

2.4 An Incremental OCL Compiler for Modeling Environments by Tamás Vajk and Tihamer Levendovszky

In software engineering, reliability and development time are two of the most important aspects, therefore, modeling environments, which aide both, are widely used during software development. UML and OCL became industry standards, and are supported by many CASE tools. OCL code checking, which has to be performed by these tools, has a specialty, as not all of the information necessary for compilation is available from the code, the related model contains the types, navigations and attributes. The build time of OCL code fragments is increased if the development tool supports distributed modeling, because in this case, model element checking has to be performed in a model repository that cannot be held in memory. In this paper, we introduce a method that enables incremental OCL code building and therefore reduces the development time. Incremental builds require higher complexity than simple builds, thus balancing between the two methods is also considered.

2.5 Implementing Advanced RBAC Administration Functionality with USE by Tanveer Mustafa, Karsten Sohr, Duc-Hanh Dang, Michael Drouineaud and Stefan Kowski

Role-based access control (RBAC) is a powerful means for laying out and developing higher-level organizational policies such as separation of duty, and for simplifying the security management process. One of the important aspects of RBAC is authorization constraints that express such organizational policies.

While RBAC has generated a great interest in the security community, organizations still seek a flexible and effective approach to impose role-based authorization constraints in their security-critical applications. In particular, today often only basic RBAC concepts have found their way into commercial RBAC products; specifically, authorization constraints are not widely supported. In this paper, we present an RBAC administration tool that can enforce certain kinds of role-based authorization constraints such as separation of duty constraints. The authorization constraint functionality is based upon the OCL validation tool USE. We also describe our practical experience that we gained on integrating OCL functionality into a prototype of an RBAC administration tool that shall be extended to a product in the future.

2.6 Shortcomings of the Embedding of OCL into QVT ImperativeOCL by Fabian Büttner and Mirco Kuhlmann

MOF QVT introduces ImperativeOCL as an imperative language for operational descriptions of model transformations (QVT operational mappings). ImperativeOCL extends conventional OCL by expressions with side-effects. A couple of semantical problems arise from the way OCL is embedded into ImperativeOCL imperative expressions are modelled as a subtype of OCL expressions. This paper points out these semantical problems and proposes a change to the operational mappings language of QVT that resolves these problems, following an approach that reuses OCL by composition rather than by inheritance in the abstract syntax of ImperativeOCL. The proposed change reduces the complexity of the imperative language, removes undefinedness, and leaves OCL conformant to its original definition.

2.7 Observations for Assertion-based Scenarios in the context of Model Validation by Emine Aydal, Richard Paige and Jim Woodcock

Certain approaches to Model-Based Testing focus on test case generation from assertions and invariants, e.g., written in the Object Constraint Language. In such a setting, assertions and invariants must be validated. Validation can be carried out via executing scenarios wherein system operations are applied to detect unsatisfied invariants or failed assertions. This paper aims to improve our understanding of how to write useful validation scenarios for assertions in OCL. To do so, we report on our experiences during the creation and execution of 237 scenarios for validating assertions for the Mondex Smart Card application. We also describe key factors that must be considered in transforming scenarios into test cases.

2.8 Executing Underspecified OCL Operation Contracts with a SAT Solver by Matthias P. Krieger and Alexander Knapp

Executing formal operation contracts is an important technique for requirements validation and rapid prototyping. Current approaches require additional guid-

ance from the user or exhibit poor performance for underspecified contracts that describe the operation results non-constructively. We present an efficient and fully automatic approach to executing OCL operation contracts which uses a satisfiability (SAT) solver. The operation contract is translated to an arithmetic formula with bounded quantifiers and later to a satisfiability problem. Based on the system state in which the operation is called and the arguments to the operation, an off-the-shelf SAT solver computes a new state that satisfies the postconditions of the operation. An effort is made to keep the changes to the system state as small as possible. We present a tool for generating Java method bodies for operations specified with OCL. The efficiency of our method is confirmed by a comparison with existing approaches.

2.9 How My Favorite Tool Supporting OCL Must Look Like by Dan Chiorean, Vladiela Petrascu and Dragos Petrascu

This paper presents its authors' viewpoint on the assessment of tools that support the use of OCL. At this time, deciding on which such tool to use is not an easy task. This is influenced by a number of objective factors, including: the user's needs, knowledge of existing tools, knowledge of the language and of the various possibilities of using it. Undoubtedly, the choice of a particular tool does also include subjective aspects. The paper is limited to the presentation of objective criteria. In this context are examined: the features that distinguish OCL within the modeling languages' family, some aspects that are either incompletely or ambiguously described in the OCL specification, the main functionalities that an OCL supporting tool should implement, the universe of tools supporting OCL. In the end, the tools listed in Section 6 are characterized with respect to the functionalities of an ideal tool and the obtained conclusions are presented.

3 Tool Showcase

The following tools were demonstrated during a two hour session at the end of the workshop: RoclET[4], The Epsilon Validation Language[5], The HOL-OCL system [6], UMLtoCSP[7], The Eye Secure Software (ESS[8]), UML Specification Environment (USE[9]), RestrictED[10], Advanced OCL Editor based on Eclipse

[4] http://www.roclet.org/ by Thomas Baar, Cédric Jeanneret and Slavisa Markovic.
[5] http://www.eclipse.org/gmt/epsilon/ by Dimitrios Kolovos and Richard Paige.
[6] http://www.brucker.ch/projects/hol-ocl/ by Achim D. Brucker and Burkhart Wolff.
[7] http://gres.uoc.edu/UMLtoCSP/ by Robert Clarisó, Jordi Cabot and Daniel Riera.
[8] http://maude.sip.ucm.es/eos/ by Miguel A. García de Dios, Marina Egea and Manuel Clavel .
[9] http://www.db.informatik.uni-bremen.de/projects/USE/ by Fabian Büttner.
[10] http://www.inf.tu-dresden.de/~ms72/RestrictED/ by Mirko Seifert and Roland Samlaus.

OCL[11], Visual Modeling and Transformation System (VMTS[12]), OCLE[13] and Kermeta[14]. Most of these demonstrations have been recorded by specialized screen capture software and can be downloaded from the workshop website[15].

4 Lessons Learned

The workshop triggered a series of interesting discussion threads, most of which may guide future work on OCL related tools. First of all, several questions and remarks related to whether OCL tools should be extended with OCL specific algorithms (analyses, optimizations, ...) or whether one should translate OCL into other languages for those purposes. For example, in the context of the performance evaluation of Clavel et al., several attendees proposed to leverage existing work on query optimization from the database domain. Others were convinced that several applications required optimizations that could only be realized within the evaluator of an OCL tool itself. In the end, it remained an open issue whether it would be most promising to focus on *bridging technological spaces* or on *improving the OCL/modeling space*, perhaps by reusing (probably specializing) expertise from other spaces.

In another discussion thread, one questioned whether or not the OCL community was mature enough to focus on performance. In this case, Jouault demonstrated that the semantics of the OCL collection operations was clear enough to reason about correctness while internally changing the mutability of collections for performance reasons. The participants agreed that one should not extend the OCL with performance specific language constructs: the language should stimulate conceptual modeling, and a modeler should not have a particular tool in mind. Obviously, it may be much easier for tool builders to expect from users that a particular style of specification is used. However, tool builders should go further and rewrite specifications that focus on non-technical issues into more technical OCL specifications that are ready for efficient execution.

Finally, the participants expressed concerns about communication channels to the authors of the OCL standard. More specifically, the community wants a light-weight communication channel to discuss problems about the standard. Summarizing such problems solely in scientific articles may be insufficient and sometimes even inappropriate.

---

[11] http://squam.info/oclеditor/ by Joanna Chimiak-Opoka, Ekrem Arslan, Hannes Moesl and Franz-Josef Peer (and presented by Michael Felderer.)
[12] http://vmts.aut.bme.hu/ by Tamas Vajk, Gergely Mezei and Tihamer Levendovszky.
[13] http://lci.cs.ubbcluj.ro/ocle/ by Dan Chiorean.
[14] http://www.kermeta.org/ by Jean-Marie Mottu.
[15] http://www.fots.ua.ac.be/events/ocl2008/