Ninth International Workshop on The Pragmatics of OCL and other textual specification languages

Jordi Cabot[1], Joanna Chimiak–Opoka[2], Martin Gogolla[4], Frédéric Jouault[3], and Alexander Knapp[5]

[1] Open University of Catalonia, Spain jcabot@uoc.edu
[2] University of Innsbruck, Austria joanna.opoka@uibk.ac.at
[3] University of Bremen, Germany gogolla@informatik.uni-bremen.de
[4] INRIA-cole des Mines de Nantes, France frederic.jouault@inria.fr
[5] University of Augsburg, Germany knapp@pst.ifi.lmu.de

Abstract. This paper reports on the 9th OCL workshop held at the MODELS conference in 2009. The workshop focused on the challeges of using OCL in a variety of new scenarios (e.g., model verification and validation, code generation, test-driven development, transformations) and application domains (e.g., domain-specific languages, web semantics) in which OCL is now being used due to the increasing popularity of model-driven development processes and the important role OCL play in them. The workshop included sessions with paper presentations and a final round discussion.

1 Introduction

In recent years, model-driven methodologies, approaches and languages (like QVT) emphasized the role that OCL has to play in all kinds of model-based software development. Now, OCL is being used for quite different purposes (e.g., model verification and validation, code generation, test–driven development, transformations) and application domains (domain–specific languages, web semantics). Successfully adapting OCL to all these different scenarios requires strong research on alternative notations/representations for OCL that simplify its application, improvements and clarifications on its semantics, modular language extensions, new evaluation techniques, mappings from OCL to other languages and formalisms and of course, the tools that will make all of this possible among many other relevant aspects.

The papers presented in the workshop covered many of these interesting topics. All submitted papers were reviewed by three industrial or academic members from the Program Committee:

- Thomas Baar, Tech@Spree, Germany
- Achim Brucker, SAP, Germany
- Manuel Clavel, IMDEA Software Institute, Spain

- Dan Chiorean, University of Cluj, Romania
- Tony Clark, Thames Valley University, 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
- Tihamer Levendovszky, Vanderbilt University, USA
- Richard Mitchell, Inferdata, UK
- Richard Paige, University of York, UK
- Mark Richters, Astrium Space Transportation, Germany
- Shane Sendall, IBM, Switzerland
- Pieter Van Gorp, University of Eindhoven
- Burkhart Wolff, LRI, University Paris–Sud, France
- Steffen Zschaler, Lancaster University, UK

2 Workshop Papers

Ten papers were accepted for presentation in the workshop. In the following, we list all papers and give a short overview of each paper from the abstracts provided by the authors (except for the two papers selected as best workshop papers, already included in this same volume in full length). Polished versions of the workshop papers will appear in an EASST special issue.

2.1 Extending OCL with Null–References: Towards a Formal Semantics for OCL 2.1 by Achim D. Brucker, Matthias P. Krieger and Burkhart Wolff

Full paper is published in this volume.

2.2 On Better Understanding OCL Collections -or- An OCL Ordered Set is not an OCL Set by Fabian Buettner, Martin Gogolla, Lars Hamann and Mirco Kuhlmann

Full paper is published in this volume.

2.3 Checking unsatisfiability for OCL constraints by Manuel Clavel, Marina Egea and Miguel Angel Garcia de Dios

In this paper we propose a mapping from a subset of OCL into first–order logic (FOL) and use this mapping for checking the unsatisfiability of sets of OCL constraints. Although still preliminary work, we argue in this paper that our mapping is both simple, since the resulting FOL sentences closely mirror the original OCL constraints, and practical, since we can use automated reasoning tools, such as automated theorem provers and SMT solvers to automatically check the unsatisfiability of non–trivial sets of OCL constraints.

2.4 A Metamodel Based Approach for Inconsistency Detection by Mohammad Ariyan, Jafar Habibi and Ali Kamandi

One of the important issues in software modeling languages is consistency management inside models. Although UMl as a standard for software modeling has been very successful compared to its predecessors, inconsistency is one of its unresolved problems. In recent years, much work has been done to detect inconsistencies in models of a language (e.g. UML), but limited research has been done to detect inconsistencies in the metamodel level. This paper presents a method for discovering possible inconsistencies at the modeling language design level. The precise semantics of the metamodel along with a target (possibly inconsistent) model are the inputs to this method. We define a translation of the input into first order predicate logic and logical reasoning is used to detect possible inconsistencies. The method will help modelers prevent modeling inconsistencies.

2.5 OCL contracts for the verification of model transformations by Eric Cariou, Belloir Nicolas, Franck Barbier and Nidal Djemam

A model–driven engineering process relies on a set of transformations which are usually sequentially executed, starting from an abstract level to produce code or a detailed implementation specification. These transformations may be entirely automated or may require manual intervention by designers. In this paper, we propose a method to verify that a transformation result is correct with respect to the transformation specification. This method both includes automated transformations and manual interventions. For that, we focus on transformation contracts written in OCL. This leads to making the proposed method independent of modeling and transformation tools. These contracts are partially or fully generated through a dedicated tool.

2.6 Requirements Analysis for an Integrated OCL Development Environment by Joanna Chimiak–Opoka, Birgit Demuth, Darius Silingas and Nicolas F. Rouquette

An Integrated OCL Development Environment (IDE4OCL) can significantly improve the pragmatics and praxis of OCL. We present the domain concepts, toollevel interactions with OCL and the use cases we identified in a systematic analysis of requirements for an IDE4OCL. The domain concepts is an important contribution of our work as it attempts to clarify inconsistencies in the relevant specifications. Because OCL is not a standalone language, the OCL landscape includes several interacting tools including an IDE4OCL. The use cases describe our vision of the desired functionality unique to an IDE4OCL. The results of our analysis and the long term vision of our work should be relevant to developers of OCL tools as well as to the OMG Request for Information regarding the UML Futures5. Our work is relevant to the UML Futures Roadmap because providing OCL for the constraints in the UML specification has been a longstanding problem at the OMG.

2.7 Generation of Formal Model Metrics for MOF based Domain Specific Languages by Christian Hein, Marcus Engelhardt, Tom Ritter and Michael Wagner

The assessment of quality in a software development process is vital for the quality of the final system. A number of approaches exist, which can be used to determine such quality properties. In a model–driven development process models are the primary artifacts. Novel technologies are needed in order to assess the quality of those artifacts. Often, the Object Constraint Language (OCL) is used to formulate model metrics and to compute them automatically afterwards. This paper describes an approach for the generation of model metrics expressed as OCL statements based on a set of generic rules. These rules can be applied on any domain specific modeling languages for creating a basic set of metrics which can be tailored for the specific needs of a development process. The paper also briefly describes a prototype of a tool for the generation, computation, and management of these model metrics by using the Software Metrics Meta–model — SMM.

2.8 A MOP Based DSL for Testing Java Programs using OCL by Tony Clark

OCL is used to specify systems by defining pre and postconditions for class operations. Typically, the conditions refer to properties and operations that are defined in a model. When the model is implemented, various implementation decisions are made regarding properties and operations that cause the OCL conditions to be inconsistent with the implementation. This paper defines a domain specific language (DSL) for testing and shows how a meta–object–protocol for OCL can be used to dynamically run tests written in the DSL against different Java implementations of the same model.

2.9 Declarative Models for Business Processes and UI Generation using OCL by Jens Brüning and Andreas Wolff

This paper presents an approach to model business processes and associated user interfaces in a declarative way, relying on constraints. An UML–based meta–model to define processes, activities and user–interface objects is proposed. Connecting activities and user interface objects in an integrated model allows expressing interdependencies and mutual effects. Flexible execution logic for workflows and UI control flows are specified by OCL invariants. The model is constructed for the UML tool USE. Using object snapshots, USE can animate and validate business scenarios. Snapshots represent states of a process and a UI at specific times. Such animation enables business process and UI designers to discuss sensible scenarios on basis of the flexible declarative models. The intention is to create validated concrete process models in connection with UI elements that will provide a basis for the system implementation.

2.10 Specifying OCL Constraints on Process Instantiations by Peter Killisperger, Markus Stumptner, Georg Peters and Thomas Stueckl

Due to the variety of concerns affecting software development in large organisations, processes have to be adapted to project specific needs to be effectively applicable in individual projects. We describe a project aiming to provide tool support for this individualised instantiation of reference processes, based on an OCL–based specification of instantiation operations. The aim is not only to execute instantiation decisions made by humans but to automatically ensure correctness of the resulting process, potentially resulting in followup actions being executed or suggested.

3 Lessons Learned

During the workshop a broad spectrum of topics was discussed: syntax, semantics and pragmatics of the language, using formal or lightweight approaches, discussing core OCL aspects or its application in different domains and presented by a mixture of academic and industrial participants. Several papers pointed out semantic inconsistencies with the current standard specification of the language and presented concrete solutions to those problems (papers 2.1, 2.2 and 2.6).

This triggered a very interesting discussion on the OCL standardization process and on how the OCL workshop community could have a more tight cooperation with the standardization committee to increase the desired community–driven character of the OCL language and thus **improve quality of the OCL specification** by opening a communication channel to easily report on the issues and improvements discussed during the workshop.

A second set of papers showed once again the broad application scenarios of OCL (this was also observed in other workshops and the main conference). Papers in the workshop presented the use of OCL as constraint or query language (papers 2.3 and 2.4) at the user model or metamodel levels (either in the context of MOF, paper 2.7, or as part of domain specific languages) and its usefulness as part of testing activities (paper 2.8), model transformations (paper 2.5) and business process modelling (papers 2.9 and 2.10).

This lead to a long discussion regarding tool support for OCL, a key aspect in all application scenarios. In this context, an initiative for a community–driven specification and development of a new user–friendly OCL tool was presented (paper 2.6). As part of discussion, attendees were asked to fill in questionnaires regarding the most important features a user–friendly OCL tool should have. Everybody agreed on the importance of such initiative and was eager to propose new features for the tool to-do list (or better said wish list). Architectural aspects and possible strategies for a collaborative-building effort of such a tool were also discussed.