Electronic Communications of the EASST
Volume 36 (2010)

Proceedings of the Workshop on OCL and Textual Modelling (OCL 2010)

Preface

3 pages

Guest Editors: Jordi Cabot, Tony Clark, Manuel Clavel, Martin Gogolla
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Preface

This contribution reports on the 10th OCL Workshop held at the MODELS conference in 2010. The workshop's motivation was to bring together researchers and practitioners in textual modelling standards, such as OCL, to report advances in the field, to share results, to identify common areas and potential for integration, and to identify common tools for developing textual modelling languages, with a view to advancing the state-of-the art. The workshop included sessions with paper presentations and a final discussion session.

Modelling started out with UML and its precursors as a graphical notation. However, graphical notations were found to have limitations in terms of specifying detailed aspects of a system design and in terms of processing and managing models. Limitations in using graphical languages include: specifying detailed behaviour; linking models to other traditional languages; making models executable; model transformation; extensions to modelling languages; model management. Many of these limitations have been addressed in recent years by proposals for textual modelling languages (e.g. there is a growing number of tools to textually define UML models) that either integrate with or replace graphical notations for modelling. Typical examples of such languages are OCL, textual MOF, Epsilon, Alloy, etc.

The current textual modelling landscape offers many interesting topics for research and experimentation including (but not limited to): new and/or successful applications; mappings to other languages/formalisms; new algorithms; evaluation strategies and optimizations for validation, verification and testing, model transformation and code generation, metamodeling/DSLs, and query and constraint specifications; alternative graphical/textual notations; evolution, transformation and simplification of expressions; libraries, templates and patterns; complexity results, quality models and benchmarks for comparing and evaluating tools and algorithms; case studies on industrial applications; experience reports; empirical studies about the benefits and drawbacks; and innovative tools. The papers presented in the workshop covered many of the aforementioned topic of interests. All submitted papers were reviewed by three industrial or academic members from the Program Committee:

- Michael Altenhofen, SAP, Germany;
- Thomas Baar, Tech@Spree, Germany;
- Mariano Belaunde, Orange Labs, France;
- Achim Brucker, SAP, Germany;
- Roberto Clarisó, Universitat Oberta de Catalunya, Spain;
- Dan Chiorean, University of Cluj, Romania;
- Joanna Chimiak-Opoka, University of Innsbruck, Austria;
- Birgit Demuth, Technical University of Dresden, Germany;
- Robert France, University of Fort Collins, USA;
- Miguel García, University of Hamburg-Harburg, Germany;

- Geri Georg, Colorado State University, USA;
- Heinrich Hussmann, University of Munich, Germany;
- Alexander Knapp, University of Augsburg, Germany;
- Tihamer Levendovszky, Vanderbilt University, USA;
- Laurent Goubet, Obeo, France;
- Richard Paige, University of York, UK;
- Mark Richters, Astrium Space Transportation, Germany;
- Shane Sendall, Snowie Research SA, Switzerland;
- Pieter Van Gorp, University of Eindhoven;
- Burkhart Wolff, LRI, University Paris-Sud, France; and
- Steffen Zschaler, Lancaster University, UK.

Two workshop papers were selected and were published in the additional LNCS volume of the MODELS 2010 conference: Integrating OCL and Textual Modelling Languages by Florian Heidenreich, Jendrik Johannes, Mirko Seifert, Michael Thiele, Christian Wende and Claas Wilke, and A Specification-based Test Case Generation Method for UML/OCL by Achim D. Brucker, Matthias P. Krieger, Delphine Longuet and Burkhart Wolff.

The workshop concluded with a discussion of features that the workshop participants would like to see in a future version of OCL. The key points in the discussion were as follows:

User Defined Parameterized Types OCL contains types such as Set(T). However there is no way for the user to define such types which would be useful to capture polymorphic structures.

Functions OCL contains many features that are similar to functional languages such as ML and Scheme. Iterators involve processing that is traditionally performed by anonymous functions in FP; however iterators are limited in comparison. Adding anonymous functions (closures) to OCL would remove the limitations and make OCL much more expressive.

Overloading Operators in OCL cannot be overloaded with respect to the type of the operands. Providing a mechanism for defining operator overloading and dynamic dispatch would make OCL more expressive.

Stereotypes OCL cannot access or define stereotypes in UML. Since stereotypes are part of UML that needs to be constrained using OCL, this situation should be addressed.

Implicit Collection Operations OCL inserts implicit `asSet` and `collect` operations when multiple links are traversed. The point was raised that this can cause confusion (as can the difference between '$->$' and '.') and makes tooling difficult. No consensus was reached

on this, but the proposal was made to force the operations to be given explicitly or to introduce different versions of these operations. It would help if tooling gave smart typing advice as expressions were typed.

Equality OCL provides a single equality operations whereas many languages provide both *identity* and *structural equivalence*. These should be added to OCL.

Reflection OCL should have better support for reflection. It should be able to reason about its own meta-definition.

Syntax OCL should have a concrete to abstraction syntax mapping.

Types The standard definition of OCL should include a type construction algorithm.

Frame Condition It is often the case that a specification needs to define a state change and also require that *everything else stays the same*. Currently OCL requires all state in scope to be explicitly referenced in a state change. It would be useful to have a frame condition such as *modifies only:* and to leave all other state in scope unchanged.

Tool Checking For OCL versions later than OCL 2.2, each chapter should be defined by a tool checked model. It should be possible to have a clear separation between an OCL core with no casting and an upper-layer with appropriate syntax for casting. The specification should define a mapping from the upper layer to the lower layer, and tools must implement this.

The workshop organizers are grateful to authors, participants, program committee members, and additional referees for their work and their contributions.

Jordi Cabot, Tony Clark, Manuel Clavel, Martin Gogolla
March 2011