On the Need for a Precise OCL Semantics

– Position Statement –

Mark Richters and Martin Gogolla

University of Bremen, FB 3, Computer Science Department
Postfach 330440, D-28334 Bremen, Germany
`{mr|gogolla}@informatik.uni-bremen.de`,
WWW home page: http://www.db.informatik.uni-bremen.de

1 The Role of OCL

The OCL is an important and valuable tool for specifying constraints on UML models. The language is to a large degree well-defined with respect to its underlying notational and semantical concepts. The notation resembles object-oriented programming languages, and the semantics is based on predicate logic. However, while the designers of OCL probably had a predicate logic semantics in mind it has yet not been documented by a precise specification. Rather the OCL is defined in a semi-formal way by using English text descriptions, a context-free grammar specifying the concrete syntax of OCL and many examples illustrating the intended meaning of expressions [4]. While this style of presentation is perfectly well-suited to introduce and demonstrate the concepts of OCL, we argue that a thorough understanding of OCL semantics requires a formal definition. The authors themselves emphasize that currently no complete formal semantics exists [3]. Due to this semi-formal nature of the OCL definition some issues have already been identified that may lead to problems or can be interpreted in different ways [1, 2].

In our view a formalization of the OCL is beneficial for achieving the following goals: (1) Improvement of the OCL itself by pointing out some potential problems with its current definition, (2) a more precise understanding of UML class models and their interpretation, (3) a solid foundation for implementing CASE tools supporting analysis, simulation and validation of UML models. We also feel that there is one aspect of OCL which often gets underrated. OCL expressions cannot only be used to specify constraints, but more generally, one can utilize them to specify queries retrieving non-trivial information about certain objects in a given system state. These queries can then be translated into statements of a query language of the chosen implementation environment. A precise OCL semantics would obviously facilitate a more systematic description of a translation scheme from OCL to e.g. SQL or the ODMG's Object Query Language (OQL).

2 Towards a Precise Semantics of OCL

For a clear understanding of OCL and as a prerequisite for building tools supporting the UML modeling process we think that the following modifications are necessary:

– *Completeness of concepts:* There should be full support of all UML language concepts available for class diagrams. For example, a test for existence of an n-ary association link currently can not be expressed with OCL.
– *Usability:* Some concepts introduced for notational convenience can produce ambiguous expressions, e.g. flattening of collections and the shortcut notation for `collect`.
– *Analysis:* Some OCL concepts need to be revisited for allowing efficient analysis or execution of constraints. Undecidable constructs should be avoided wherever possible.
– *Data versus objects:* The distinction between data and objects should be made clearer. Objects have a state that operations may depend on whereas data values are stateless.

In [6] we have addressed some of these issues and presented a formalization of OCL and certain aspects of UML class models. We have defined syntax and semantics of OCL expressions and have given examples for their evaluation.

3 Automated Support for OCL Expression Analysis

A CASE environment supporting rigor development with UML should provide two fundamental facilities for checking a model: (1) verification against *formal* requirements, and (2) validation against *informal* user requirements. We concentrate on validation in the following. Validation can be supported e.g. by rapid prototyping or by simulating the expected behavior of the system. In the past, we had some good experience with an animation system for "executing" specifications written in the object specification language TROLL *light* [5].

We argue that an animation system for UML should be based on a well-defined semantics for UML and – in particular – OCL. Modelers use OCL to constrain the set of valid system states of a system specified with UML. OCL constraints may be complex and not always easy to understand in their relationship to other constraints. Animation of a UML model means creation, manipulation, and destruction of specified objects under user control. All these operations are only allowed if they do not violate any constraints. The process of observing real object behavior quickly reveals whether some constraints are e.g. too permissive, too restrictive or not even adequate. This will help making modifications to the UML model eventually improving the quality of the system to be developed.

Some conceptual and technical problems have to be solved before OCL can successfully be implemented in an animation system. First, one has to make sure that the evaluation of expressions always terminates. One way to achieve this is by carefully distinguishing between finite and infinite sets of objects or values. Also, the distinction between data and objects should be made clearer. By allowing only a finite number of objects in any system state the quantification and evaluation of OCL expressions in principle becomes decidable.

4 Defining OCL Semantics with a Metamodel

A precise definition of the semantics of OCL expressions firstly requires a precise definition of OCL syntax. The OMG standard describes the concrete syntax of OCL but a semantics definition usually builds upon the abstract syntax of a language. There are different ways to describe the abstract syntax. In [7] we used a metamodeling approach that nicely fits within the UML metamodeling architecture.

Our OCL metamodel delivers a more precise and detailed view of OCL. As a consequence from using only well-known modeling concepts of UML which are compliant with the Meta Object Facility (MOF), our OCL metamodel can easily be read by everybody familiar with UML. The decomposition into packages for types, expressions, and values emphasizes the basic structure of OCL. Some issues that remained unclear from the original documentation had to be resolved and solutions for these issues have been integrated into the metamodel. For example, we have presented a concrete approach for connecting the OCL with the UML core concepts. Furthermore, we have given an interpretation for several important OCL concepts, e.g. the context of an expression, undefined values, and result values of expressions.

Although the metamodel provides a precise description of the abstract syntax of OCL expressions, a complete description of OCL still requires a proper semantics definition. In our framework, a mapping from expressions to values would be necessary. It would be interesting to see, whether the UML modeling features are sufficient to express the OCL semantics within the metamodel.

5 Conclusion

To summarize our position, we think that the OCL is of great benefit for all modelers striving for more precise UML design models. The definition of a formal semantics for OCL complements the OMG standard and certainly facilitates the development of automated tool support for OCL.

References

1. Martin Gogolla and Mark Richters. Equivalence rules for UML class diagrams. In Pierre-Alain Muller and Jean Bézivin, editors, *Proceedings of UML'98 International Workshop, Mulhouse, France, June 3 - 4, 1998*, pages 87–96. ESSAIM, Mulhouse, France, 1998.
2. Ali Hamie, Franco Civello, John Howse, Stuart Kent, and Richard Mitchell. Reflections on the object constraint language. In Pierre-Alain Muller and Jean Bézivin, editors, *Proceedings of UML'98 International Workshop, Mulhouse, France, June 3 - 4, 1998*, pages 137–145. ESSAIM, Mulhouse, France, 1998.
3. Anneke Kleppe, Jos Warmer, and Steve Cook. Informal formality? the object constraint language and its application in the UML metamodel. In Pierre-Alain Muller and Jean Bézivin, editors, *Proceedings of UML'98 International Workshop, Mulhouse, France, June 3 - 4, 1998*, pages 127–136. ESSAIM, Mulhouse, France, 1998.
4. OMG. Object Constraint Language Specification. In OMG, editor, *OMG Unified Modeling Language Specification, Version 1.3, June 1999*, chapter 7. Object Management Group, Inc., Framingham, Mass., Internet: http://www.omg.org, 1999.
5. Mark Richters and Martin Gogolla. A Web-based Animator for Validating Object Specifications. In Bipin C. Desai and Barry Eaglestone, editors, *IDEAS'97: International Database Engineering & Applications Symposium, Montreal, Canada, August 25–27, 1997*, pages 211–219, 1997.
6. Mark Richters and Martin Gogolla. On formalizing the UML object constraint language OCL. In Tok Wang Ling, Sudha Ram, and Mong Li Lee, editors, *Proc. 17th Int. Conf. Conceptual Modeling (ER'98)*, pages 449–464. Springer, Berlin, LNCS 1507, 1998.
7. Mark Richters and Martin Gogolla. A metamodel for OCL. In Robert France and Bernhard Rumpe, editors, *Proceedings of UML'99 International Workshop, Fort Collins, Colorado, USA, Oct 28–30, 1999*, 1999.