MOVA: A Tool for Modeling, Measuring and Validating UML Class Diagrams

Manuel Clavel, Marina Egea, and Viviane Torres da Silva

Universidad Complutense de Madrid, Spain
clavel@sip.ucm.es, marina_egea@fdi.ucm.es, viviane@fdi.ucm.es

The MOVA tool is a rewriting-based UML [8] modeling, measuring, and validation tool developed as part of a broader effort for integrating rigorous modeling and validation into the software engineering curricula and the industrial software engineering process.

Validation and testing in software development have been recognized of key importance for long. For certain aspects of a design, UML class diagrams do not provide the level of conciseness and expressiveness which a textual language can offer. The Object Constraint Language (OCL) [7] is the standard language for writing constraints in order to disambiguate, clarify, and precise the meaning of UML models. In this context, validating a UML class diagram reduces to checking whether its prototypical instances fulfill the desired OCL constraints. A number of CASE tools exists which facilitate drawing and documenting UML diagrams. However, they provide little support for writing constraints in OCL and generally no support for validating UML class diagrams. The MOVA tool has been designed, primarily, as a validating tool: as such, it provides advance support for:

– writing OCL constraints over UML class diagrams;
– drawing instances of UML class diagrams; and
– checking OCL constraints over instances of UML class diagrams.

Software metrics have been widely used to improve productivity and quality during the software development life-cycle. Metrics have been applied to the software design, to the software implementation, and also to the software development process itself. The early use of metrics in the software life-cycle can provide quantitative indicators and predictors of structural problems [1]. Model metrics can be written in OCL [2]: they are queries to be evaluated over the instances of the metamodel that correspond to the models being measured. The MOVA tool is also a measuring tool: as such, it provides support for:

– writing metrics over the MOVA metamodel; and
– evaluating metrics over the UML class diagrams.

The MOVA tool is still an experimental tool. Despite its limitations, it has been successfully used in software engineering courses: students appreciate the friendliness of the GUI interface when drawing class and object diagrams, and of the OCL editor when writing and evaluating constraints. The latest version of the tool, along with its user manual and a collection of examples, is available at http://maude.sip.ucm.es/mova.

The MOVA Tool

MOVA from the outside The MOVA tool allows the user to draw UML class and object diagrams, write and check OCL invariants, write and evaluate OCL queries, write and evaluate OCL metrics, and define OCL operations to be used in invariants, queries, and metrics. For example, Figure 1 shows a class diagram TRAIN-WAGON, which models a simple railway system, along with an object diagram TRAIN-WAGON-1, which models one of its instances.

The MOVA OCL editor assists the user in writing OCL expressions in three different contexts: adding a constraint to a class diagram; writing a query about an object diagram; and defining the body of an operation. The editor includes a model-based syntax-guiding facility which is unique, to the best of our knowledge, among the non-commercial available tools [3, 9, 11, 10, 6]. The editing facilities provided by other tools typically amount to parsing, type-checking, and uncontextualized syntax-highlighting. With the MOVA editor, to write an OCL expression the user selects patterns from lists that are built at "run time" when the buttons Start, Dot, Arrow, or Space are pressed. The actual patterns shown in the lists depend on the model under consideration, the current type of the expression, and the button that has been pressed: the Start button is used to start writing an expression; the Dot button is used to access a property of a class; the Arrow button is used to access a property of a collection; and the Space button is used to introduce a logical or an arithmetic operator. Patterns can contain "holes". When a pattern with holes is selected the user has to fill its holes with expressions of the appropriate type. Each of these expressions is written in a new editor window which is a child of the window in which the pattern has been selected. For example, Figure 2 shows the sequence of patterns that are selected in order to add the following constraint to railway model TRAIN-WAGON:

context Train **inv:** AtLeastOneSmokingWagon
self.wagon→exists(w|w.smoking)

The MOVA metrication facility, which allows the user to define and execute his/her own metrics directly over the models, is also unique among the available tools. To the best of our knowledge MOVA is the only available modeling tool that allows its users to write and execute their own metrics directly over the user-models. To help the users in writing and executing their own metrics, MOVA offers the OCL editor and an internal translator from MOVA user-models to their corresponding MOVA metamodel instances. Recall that metrics in MOVA are simply OCL queries over the object diagrams which model the instances of the MOVA metamodel.

MOVA from the inside The MOVA tool is a Java IDE for the ITP/OCL tool, a text-input UML modeling and validation tool. The ITP/OCL tool provides commands for building class and object diagrams, validating OCL constraints, and evaluating OCL queries. The ITP/OCL tool relies on the membership equational semantics for UML class diagrams with OCL constraints proposed in [5]. The ITP/OCL tool is written entirely in Maude [4], a rewriting-based programming language that implements membership equational logic. Events on the MOVA graphical interface are transformed into ITP/OCL's text-input commands and are interpreted and executed in a Maude process running the ITP/OCL tool.

Future work

In its current state, the MOVA tool suffer of three main shortcomings. Firstly, the MOVA metamodel is only a subset of the UML metamodel. In particular, packages and non-query operations are not specified and, consequently, it is not possible to model such elements nor to define OCL expressions associated with them. Secondly, the MOVA editor does not support the full OCL syntax; in particular, let expressions, and Bag and Sequence types are not yet supported. Finally, class and object diagrams are saved in a MOVA specific XML format, which precludes the models for being exchange with other tools. To overcome these shortcomings, we plan to extend in the near future the MOVA metamodel and the MOVA OCL editor to comply with the UML and OCL standards, and the MOVA saving and loading facilities to support XMI format.

References

1. A. L. Baroni and F. Brito-Abreu. A formal library for aiding metrics extraction. In *International Workshop on Object-Oriented Re-Engineering at ECOOP 2003*, Darmstadt, Germany, July 2003.
2. F. Brito e Abreu. Using OCL to formalize object oriented metrics definitions. Technical report, FCT/UNL and INESC, 2001. Technical Report ES007/2001.
3. D. Chiorean. OCLE User Manual. http://lci.cs.ubbcluj.ro/ocle/features.htm, 2006. Provided with the distribution of OCLE.
4. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 285:187–243, 2002.
5. M. Clavel and M. Egea. Equational specification of UML+OCL class diagrams. Technical report, 2007. Manuscript. http://maude.sip.ucm.es/ clavel/pubs/.
6. S. Loecher and S. Ocke. A metamodel-based OCL-compiler for UML and MOF. In *OCL 2.0 - Industry standard or scientific playground?, Workshop Proceedings, 6th International Conference on the Unified Modelling Language and its Applications*, volume 154. ENTCS. Elsevier, October 2003.
7. Object Management Group. Object constraint language specification (2.0). Technical report, Frammingam, Mass, 2006. http://www.omg.org.
8. Object Management Group. Unified modeling language specification (2.1.1). Technical report, Frammingam, Mass, 2007. http://www.uml.org.
9. M. Richters. The USE tool : A UML-based specication environment. Technical report, 2001. http://www.db.informatik.uni-bremen.de/projects/USE/.
10. G. Vanwormhoudt, M. Payen, and Y. Bontems. EMF OCL plugin 0.1, 2004. http://www.enic.fr/people/Vanwormhoudt/siteEMFOCL/index.html.
11. J. Warmer and A. Kleppe. Octopus: OCL tool for precise UML specifications. Technical report, 2006. http://www.klasse.nl/octopus/index.html.

Screenshots

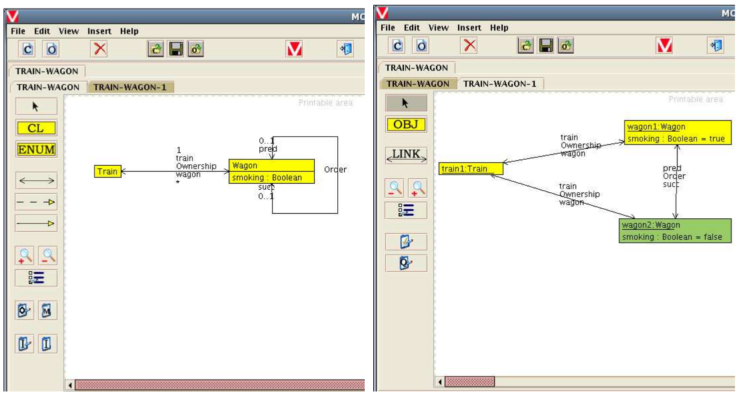

Fig. 1. A class diagram and one of its object diagrams.

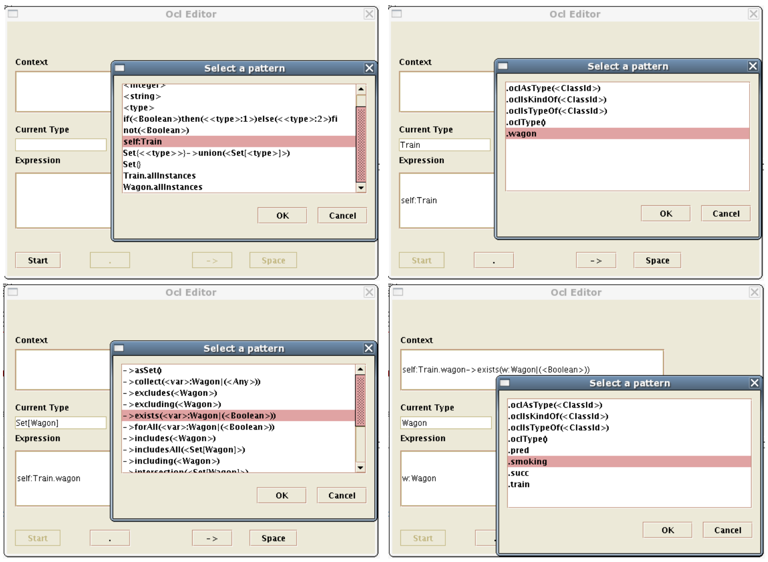

Fig. 2. A sequence of pattern selections.