Employing the Object Constraint Language in Model-Based Engineering

Martin Gogolla

Database Systems Group, University of Bremen, Germany
gogolla@informatik.uni-bremen.de

Abstract. MBE (Model-Based Engineering) proposes to develop software by taking advantage of models, in contrast to traditional code-centric development approaches. If models play a central role in development, model properties must be formulated and checked early on the modeling level, not late on the implementation level. We discuss how to validate and verify model properties in the context of modeling languages like the UML (Unified Modeling Language) combined with textual restrictions formulated in the OCL (Object Constraint Language).

Typical modeling and transformation languages like UML (Unified Modeling Language), EMF (Eclipse Modeling Framework), QVT (Queries, Views, and Transformations) or ATL (Atlan Transformation Language) are complemented by the textual OCL (Object Constraint Language) enriching graphical or textual models with necessary details. Models allow the developer to formulate essential system properties in an implementation- and platform-independent way.

Precise object-oriented development must take into account system structure and system behavior. The system structure is often captured by class diagrams and can be instantiated in terms of prototypical exemplars by object diagrams. The system behavior can be determined by statechart diagrams, and system execution traces can be demonstrated by sequence diagrams. OCL restricts the possible system states and transitions through the use of class invariants, operation pre- and postconditions, state invariants, and transition pre- and postconditions. OCL can also be used during model development as a query language.

Modeling features and their analysis through validation and verification must be supported by tools like, for example, the tool USE (UML-based Specification Environment) [1]. Within USE, UML class, object, statechart, and sequence diagrams extended with OCL are available [2]. OCL has been extended with programming language features in SOIL (Simple Ocl-like Imperative Language) which allows the developer to build operation realizations on the modeling level without having to dig into implementation level details [3]. Thus models in USE are executable, but a prescriptive SOIL model for operations can be checked against descriptive plain OCL pre- and postconditions.

Tools like USE assist the developer in order to validate and to verify model characteristics. Validation and verification can be realized, in USE for example, by employing a so-called model validator based on relational logic and SMT solvers [4]. Model properties to be inspected include consistency, redundancy freeness, checking consequences from stated constraints, and reachability [5]. These properties are handled on the conceptual modeling level, not on an implementation level. Employing these instruments, central and crucial model and transformation model characteristics can be successfully analyzed and checked.

Transformation models [6] provide an approach that formulates model transformations in a descriptive, not prescriptive way by stating desired properties of transformations in terms of input and output models and their relationship. From transformation models, tests [7] can be derived that are independent from the employed transformation language. Modeling and model transformations with USE have been successfully applied in a number of projects, for example, in the context of public authority data interchange [8].

Acknowledgement

The force and energy of Mark Richters, Jörn Bohling, Fabian Büttner, Mirco Kuhlmann and Lars Hamann formed USE as it stands today. Thanks!

References

1. Gogolla, M., Büttner, F., Richters, M.: USE: A UML-Based Specification Environment for Validating UML and OCL. Science of Computer Programming **69** (2007) 27–34
2. Hamann, L., Hofrichter, O., Gogolla, M.: Towards Integrated Structure and Behavior Modeling with OCL. In France, R., Kazmeier, J., Breu, R., Atkinson, C., eds.: Proc. 15th Int. Conf. Model Driven Engineering Languages and Systems (MoDELS'2012), Springer, Berlin, LNCS 7590 (2012) 235–251
3. Büttner, F., Gogolla, M.: Modular Embedding of the Object Constraint Language into a Programming Language. In Simao, A., Morgan, C., eds.: Proc. 14th Braz. Symp. Formal Methods (SBMF'2011), Springer, Berlin, LNCS 7021 (2011) 124–139
4. Kuhlmann, M., Gogolla, M.: From UML and OCL to Relational Logic and Back. In France, R., Kazmeier, J., Breu, R., Atkinson, C., eds.: Proc. 15th Int. Conf. Model Driven Engineering Languages and Systems (MoDELS'2012), Springer, Berlin, LNCS 7590 (2012) 415–431
5. Gogolla, M., Kuhlmann, M., Hamann, L.: Consistency, Independence and Consequences in UML and OCL Models. In Dubois, C., ed.: Proc. 3rd Int. Conf. Test and Proof (TAP'2009), Springer, Berlin, LNCS 5668 (2009) 90–104
6. Bezivin, J., Büttner, F., Gogolla, M., Jouault, F., Kurtev, I., Lindow, A.: Model Transformations? Transformation Models! In Nierstrasz, O., Whittle, J., Harel, D., Reggio, G., eds.: Proc. 9th Int. Conf. Model Driven Engineering Languages and Systems (MoDELS'2006), Springer, Berlin, LNCS 4199 (2006)
7. Gogolla, M., Vallecillo, A.: Tractable Model Transformation Testing. In France, R., Küster, J.M., Bordbar, B., Paige, R.F., eds.: Proc. 7th Int. Conf. Modelling Foundations and Applications (ECMFA'2011), Springer, Berlin, LNCS 6698 (2011) 221–235
8. Büttner, F., Bartels, U., Hamann, L., Hofrichter, O., Kuhlmann, M., Gogolla, M., Rabe, L., Steimke, F., Rabenstein, Y., Stosiek, A.: Model-Driven Standardization of Public Authority Data Interchange. Science of Computer Programming (2013)