Model Development in the UML-based Specification Environment (USE)

Martin Gogolla

University of Bremen, Computer Science Department
gogolla@informatik.uni-bremen.de

Abstract. The tool USE (UML-based Specification Environment) supports analysts, designers, and developers in executing UML models and checking OCL constraints and thus enables them to employ model-driven techniques for software production. USE has been developed since 1998 at the University of Bremen. This paper will shortly discuss to what extent and how USE relates to the selected questions and topics (like model quality or modelling method) raised for this seminar.

1 Context of USE

The Object Constraint Language (OCL) is an important part of the Unified Modeling Language (UML) [OMG05,RJB05]. OCL can be used within UML for various tasks:

- In class diagrams, invariants for classes and pre- and postconditions for operations as well as operation definitions can be described in OCL.
- In statecharts, guards of transitions and pre- and postconditions of actions may be characterized with OCL.
- Within use case diagrams, it is possible to state pre- and postconditions of use cases.
- There are many other opportunities for employing OCL within the UML.

In our view OCL is a kind of high-level assembler language for the UML, because many UML features may be translated into simpler UML features with additional OCL restrictions. In the recent years, we have developed in our group an OCL interpreter called USE (UML-based Specification Environment [GBR05]). USE covers a substantial part of UML as well.

2 Scope of USE

USE currently supports class, object, and sequence diagrams: Class invariants can be checked in system states which are represented as object diagrams. Sequence diagrams represent state manipulation and operation call sequences (scenarios) in which pre- and postconditions are checked. Operations can be realized with OCL expressions directly or in an imperative style.

System states can be described by explicit commands for creating and destroying objects and links and setting attributes. System states can also be described implicitly by giving properties and by providing a search space in a small imperative programming language [GBR05] for object diagram sequences.

System states and scenarios can be inspected in detail with OCL queries, an invariant checker, and an evaluation browser which can be applied for analysing the reasons for error occurrence.

3 Tasks for USE

Like Alloy [Jac02] USE is more a model finder than a model checker, i.e., it helps developers in finding models with desired properties. Thus, a main task of USE is to get confidence into developed models, i.e., formal descriptions, by testing it with system states (object diagrams) and scenarios (sequence diagrams). USE allows:

- to check the consistency of models (with invariants and pre- and postconditions) by constructing an object resp. sequence diagram,
- to show the independence of invariants (no invariant follows from the other invariants) by constructing a state violating the invariant under consideration but satisfying all other invariants (find a state satisfying the negated invariant and all other invariants), and
- to check whether an undesired property does hold or does not hold by showing that it is not possible to construct a state where all invariants and the unwanted property hold (this is called certifying a property); the drawback of this technique is that the search space for state construction has to be restricted explicitly by an imperative program.

4 One Model Development Method for USE

One reasonable model development method within USE could be achieved as follows:

- Start with a model possessing many global invariants and structural elements, i.e., attributes and association ends, only.
- Add behavioural elements, i.e., operations.
- Transform the global invariants by equivalence transformation into local pre- and postconditions as far as possible. Thereby replace expensive global checks by cheap local checks.
- Test your intermediate models by system states and scenarios which can constantly and incrementally be developed and extended.

This approach seems applicable at least in information system and database design.

5 Model Qualities supported by USE

The following model qualities are derived from (the intersection of) software qualities introduced in textbooks on software engineering [GJM03,Som04].

- Correctness, Reliability, Robustness: USE allows to check constraints and the system functionality. With USE positive and negative test cases can be formulated and checked.
- Performance, Efficiency: USE offers to go from global invariants to local pre- and postconditions for operations.
- Usability: USE supports checking of operation calls. By this, the ease of providing appropriate parameters for operations and larger operation sequences describing use cases can be tested.
- Portability: By supporting the idea of developing platform-independent models, USE offers support for portable systems.
- Understandability: By offering the possibility for looking at concrete system states and scenarios, USE helps in tracing complex formal descriptions.

References

[GBR05] Martin Gogolla, Jörn Bohling, and Mark Richters. Validating UML and OCL models in USE by automatic snapshot generation. *Software and System Modeling*, 4(4):386–398, 2005.

[GJM03] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering.* Prentice Hall, 2nd edition, 2003.

[OMG05] Object Management Group. *OMG Unified Modeling Language, Version 2.0.* OMG, `http://www.omg.com/uml/`, 2005.

[Jac02] Daniel Jackson. Alloy: A lightweight object modeling notation. *ACM Transactions on Software Engineering and Methodology*, 11(2):256–290, April 2002.

[RJB05] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual.* Object Technology Series. Addison-Wesley, Reading, Massachusetts, 2nd edition, 2005.

[Som04] Ian Sommerville. *Software Engineering.* Addison-Wesley, 7th edition, 2004.