Softw Syst Model (2011) 10:439–440
DOI 10.1007/s10270-011-0210-3

EDITORIAL

# Variability in UML language and semantics

**Bernhard Rumpe · Robert France**

Published online: 28 August 2011


Practitioners, who use UML as a sketching language are generally not too concerned about the precision of their models, but developers who build UML models to rigorously analyze software properties (e.g., to analyze the consistency of design constraints) or that can be mechanically transformed to implementations requiring tools and tool chains that are based on a precisely defined UML semantics (see this issue's Expert Voice by Manfred Broy and María Victoria Cengarle as well as the regular paper on the many semantics of sequence diagrams by Zoltán Micskei and Hélène Waeselynck). This need motivates much of the work on defining appropriate formal semantics for the UML.

There is a significantly large body of work on formalizing the UML—both syntactical appearance, internal representation and semantics (in terms of meaning), and the collective experience suggests that defining appropriate semantics for the UML has both a technical and a strong political/social aspect. This non-technical aspect is concerned primarily with determining what constitutes an "appropriate" language. The problem is that different stakeholders, including UML modelers from different domains, tool vendors with specific ready to use solutions, have varying views of what constitutes an appropriate UML language and its semantics.

It is not easily possible to support these sometimes competing views in a single language. This led to the view of UML as a "family of languages" and to the introduction of profile mechanisms and "semantic variation points" that can be used for specializing the syntax and semantics of UML. The UML currently has a wide variety of these semantic variation points indicating points in the language definition that can be tailored to better support the many forms of usage of UML. Although this form of tailoring may be convenient for developers, it makes the development of generic tools and tool chains considerably more complex and makes it almost impossible to provide a well formed, rather complete and precise semantics for the UML as a whole. Furthermore, the UML does currently not provide good mechanisms for introducing and describing variations or selecting concrete sub-variants yet.

Managing variability within a language, such as the UML can be likened to manage variability of a software product line. Indeed, it is useful to regard the UML as a product line of languages to explore how techniques for managing variability in product lines (e.g., feature diagrams) can be used to explicitly manage variability in UML. We recently invested some efforts in studying this technique and our work suggests that it can very well be used to make the UML, or at least some derivatives of UML, more precise and easier to use. It can also help developers understand similarities and differences across different UML derivatives. One can envisage configuring a UML tool using a configuration that describes a particular UML derivative, the required tool functionality, enhanced analysis algorithms or domain-specific restrictions, the desired form of code and test generation, among other features. One can also envisage that the UML standard defines its semantic variation points explicitly using feature diagrams.

From its many possible forms of uses, it seems clear that the UML will not have a single syntactic form or semantics that adequately serves its community, but understanding and managing variations in the language UML might allow us to cope with this drawback.

The time to explore language variability to allow modelers deal with precise and well-assisted language variations.

B. Rumpe (✉)
RWTH Aachen, Aachen, Germany
e-mail: bernhard.rumpe@sosym.org

R. France (✉)
Colorado State University, Fort Collins, CO, USA
e-mail: france@cs.colostate.edu

**Contents in this issue**

In this issue, we present one expert voice and six regular papers.

The Expert Voice on "UML Formal Semantics: Lessons Learned" where Manfred Broy and María Victoria Cengarle discusses currently known problems with the UML. They discuss problems related to formal semantics, problems with integrating UML sublanguages, and they point out that the UML standard is written from and for tool vendors and not for users, among other things.

Shahar Maoz and David Harel in their regular paper "On tracing reactive systems" present a technique for analyzing, visualizing, and exploring the execution traces of reactive systems. This paper describes how the rigorous UML2-dialect, Live Sequence Charts (LSC), can be used by developers to think and model in terms scenarios, and to produce quality code in a more efficient way.

The regular paper "Rigorous identification and encoding of trace-links in model-driven engineering" by Richard Paige, Nikolaos Drivalos, Dimitrios Kolovos, Kiran Fernandes, Christopher Power, Goran Olsen, and Steffen Zschaler discusses the complexity of tracing links in model driven engineering. The problem addressed in this paper is significant, given that the use of modeling techniques in a project can lead to the development of many model artifacts related in a variety of ways that need to be managed.

Zoltán Micskei and Hélène Waeselynck present a survey of sequence diagram semantics in their paper "The many meanings of UML 2 Sequence Diagrams: a survey". As mentioned in our editorial, a variety of semantics can be associated with UML elements. This paper is a good attempt at comparing a variety of semantics for sequence diagrams.

Although many of our papers mainly deal with languages and semantics, we also like to receive good papers that describe how modeling can be effectively applied to problems in other domains. In their regular paper "An executable object-oriented semantics and its application to firewall verification" the authors Kenro Yatake and Takuya Katayama target ML for simulation and the HOL theorem prover for verification of their semantics.

Martin Monperrus, Jean-Marc Jézéquel, Benoit Baudry, Joël Champeau, and Brigitte Hoeltzener in their regular paper "Model-driven generative development of measurement software" tackle the problem of efficiently measuring software that is developed using domain-specific languages. They present a model-driven approach to measure such models with respect to standard quality metrics.

The last regular paper "Model-based qualitative risk assessment for availability of IT infrastructures", Emmanuele Zambon, Sandro Etalle, Roel J. Wieringa and Pieter Hartel presents a model-based approach to risk assessment for IT infrastructures. The authors discuss, why classic risk assessment processes do not fit to IT and propose an improved process.

We hope you enjoy reading this issue.

Robert France, Bernhard Rumpe
Editors in Chief