Automated Software Engineering Special Issue on Precise Semantics for Software Modeling Techniques (PSMT'98 – an ICSE'98 Workshop)

Tom S. E. Maibaum
Imperial College
London SW7 2BZ, UK
+44-171-594-8274
tsem@doc.ic.ac.uk

Bernhard Rumpe
TU München
80333 München, Germany
++49-89-289-28129
rumpe@in.tum.de

November 9, 1999

1 Introduction

Over the last few years, there has been a growing recognition that the worlds of Formal Methods and the CASE tool supported modelling techniques must come together to provide Software Engineers with soundly based, but notationally familiar development environments and techniques. Since many engineering disciplines use what appear to be informal, sometimes iconic, languages as 'interfaces' to their mathematical languages for modelling application solutions, it seems plausible to try the same approach in Software Engineering. This means, effectively, that we should take extant Software Modeling Techniques and see if we can develop formal semantics for their notations, so as to provide software engineers with familiar tools, and also providing them with the possibility of performing the analyses and formal checks, on the one hand, and the support for transformational techniques being applied for implementation and code generation, on the other. (In the longer term, effective formal notations may generate modelling techniques and notations which will then be adopted by Software Engineers.)

With this motivation in mind, the PSMT workshop was organized in conjunction with ICSE'98 in Kyoto on April 20th, so as to take a critical look at recent

thoughts and developments in this emerging area. Work based on three of the submissions of this workshop have been selected for inclusion in this journal.

2 Workshop Themes

Currently there is an ongoing standardization process for syntactical representations of object-oriented modeling techniques (MT) initiated by the OMG, which had its first notable output in the standardisation of UML [2]. The standardization of MT should not only involve provision of a precise syntax, but also of a precise semantics. This is essential for an unambiguous understanding of system specifications written using a MT, escpecially when using diagrammatic and iconic languages, as they are very common in software engineering.

A precise semantics allows us to detect inconsistencies and inaccuracies both in MT themselves (metareasoning about the MT used), and in specifications written using these MT (reasoning about the system under design). It also provides a means for comparing different MT in a more precise way and for improving the notation. Furthermore, it enables precise characterisation of interoperability between different MT. From an engineering perspective, it also allows us to use a notation in a more standardized way, thus leading to better and less ambiguous understanding, supporting true reuse of specifications and designs, and a more accurate definition of context conditions or (code) generators. Also requirements decisions can be traced more precisely to code produced from them. Based on a precise semantics of modeling techniques, tool support beyond graphic editors becomes possible. Finally, the integration of tools and the combination of methods is more feasible than today.

We would like to express our immense gratitude to the rest of the Program Commitee, which consisted of Manfred Broy (TU Munich), Derek Coleman (Hewlett-Packard), Desmond D' Souza (ICON Computing), Robert France (Florida Atlantic University), Øystein Haugen (Ericsson, Oslo), and Bran Selic (ObjecTime, Ottawa). Thanks go also to the additional reviewers of the journal versions of the papers, namely Colin Atkinson, Kokichi Futatsugi, Pavel Hruby, Haim Kilov, Ulrike Lechner, Alexander Schmidt, and Wolfgang Schwerin.

In their paper *Logic of Change: Semantics of Object Systems with Active Relations* I. Bider, M. Khomyakov, and E. Pushchinsky present a new model for programming. It extends object-orientation by employing active relations. This is especially suited for business applications, where relations actively maintain business rules. A logical semantics, as well as a procedural semantics based on state machines, is given and an appropriate programming language is discussed.

T. Mens and T. D'Hondt in their paper *Automating Support for Software Evolution in UML* focus on the question of how to use UML concepts to improve the development process. They particularly concentrate on the potential of UML with respect to iterative evolution of software within and between projects. They

identify the lack of a precise semantics for UML as one of the main inhibitors that needs to be overcome and they suggest some additional features for UML that especially support evolution.

Despite its widespread use and industrial importance, SDL lacks at present a complete and integrated formal semantics. A formal semantics for SDL using a new algebraic formalism called Timed Rewriting Logic (TRL) is presented by L. J. Steggles and P. Kosiuczenko in their paper *A Formal Model for SDL Specifications based on Timed Rewriting Logic*. The given semantics provides a natural basis for analysing, verifying, testing and composing SDL systems. The authors present a new equivalence theorem that allows this TRL semantics to be automated using Rewriting Logic and its associated tools. This is demonstrated by modelling an SDL specification for the benchmark alternating bit protocol.

References

[1] M. Broy, D. Coleman, T. S. E. Maibaum, B. Rumpe. PSMT – Workshop on Precise Semantics for Software Modeling Techniques. Proceedings. Technical Report TUM-I9803, Technische Universität München, April 1998.

[2] OMG. The Unified Modeling Language (UML) Specification. Version 1.4. Technical recommendation. 1999.