PSMT – ICSE 98 Workshop on Precise Semantics for Software Modeling Techniques

Manfred Broy
TU München
80333 München, Germany
++49-89-289-28162
broy@forsoft.de

Derek Coleman
King's College
Strand, London WC2R 2LS, UK
Tel: +44-171-873-2588
derek@dcs.kcl.ac.uk

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@forsoft.de

1 Introduction

Over the last few years, it is being more and more often recognized 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, but also providing 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.

With this motivation in mind, the organisers were of the view that ICSE provided a profitable venue for a small scale meeting which could take a critical look at recent thoughts and developments in this emerging area. The sections below outline the workshop themes and topics and give brief descriptions of the discussion papers to be presented at the workshop itself.

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 1.1 [1]. A standardization of MT does not only involve a precise syntax, but also a precise semantics. This is essential for an unambiguous understanding of system specifications given by 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 produced code. Based on a precise semantics of modeling techniques, tool support beyond graphic editors becomes possible. Then, even the integration of tools and the combination of methods is more feasible than today.

Therefore, we decided to organize a workshop in conjunction with ICSE 98 in Kyoto on April 20th, to bring together researchers, requirements engineers, software engineers, and tool builders for substantial discussions related to the formalisation of MT.

3 Topics

The workshop is mainly focused around the following topics:

- Methods using formal diagrammatic/iconic MT
- How precise semantics can improve the development process
- Precise semantics for diagrammatic/iconic MT

- Integration of semantics for heterogeneous MT
- Formal development and refinement concepts for diagrammatic/iconic MT
- Comparison of existing semantic models
- Ways to achieve precision of syntax and semantics
- Tool support
- Standardizing MT

4 Workshop Submissions

From 13 submitted papers of full length seven have been accepted . The papers include new and interesting ideas and give an overview of relevant work. Papers presented at the workshop are published in this technical report, which is also available online `http://www.forsoft.de/~rumpe/psmt98-ws/`.

We would like to express our immense gratitude to all the members of the Program Commitee, which consists of

- Manfred Broy (TU Munich),
- Derek Coleman (King's College, London),
- Desmond D' Souza (ICON Computing),
- Robert France (Florida Atlantic University),
- Tom S. E. Maibaum (Imperial College, London),
- Øystein Haugen (Ericsson, Oslo),
- Bernhard Rumpe (TU Munich), and
- Bran Selic (ObjecTime, Ottawa).

Thanks go also to the additional reviewers, namely

- Radu Grosu,
- Ursula Hinkel,
- Birger Møller-Pedersen,
- Barbara Paech,
- Wolfgang Schwerin, and
- Veronika Thurner,

who did a great job.

5 Paper Overviews

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 busines rules. A logical semantics as well as a procedural semantics based on state machines is given, and an appropriate programming language is discussed.

The paper *Logical Semantics for CafeOBJ* presented by R. Diaconescu and K. Futatsugi gives a survey of the semantics of the CafeOBJ system and language. The latter is a succesor of the famous algebraic specification and programming language OBJ adding several new primitive paradigms to the traditional OBJ language, such as rewriting logic, and behavioural concurrent specification.

In their paper *State Diagrams in UML: A Formal Semantics using Graph Transformations* M. Gogolla and F. Parisi Presicce show how to transform UML (Unified Modeling Language) state diagrams into graphs by making explicit the intended semantics of the diagram. The process of state expansion in nested state diagrams is explained by graph transformations. The general idea of approaching the semantics of UML diagrams by graph transformations is applicable to other forms of UML diagrams as well. The main advantage of the graph transformation approach is the closeness between the (mathematical) graph representation and the (UML) diagram representation.

T. Mens, P. Steyaert, and C. Lucas in their paper *Giving Precise Semantics to Reuse and Evolution in UML* focus on the question, of how to use UML concepts to improve the development process. They especially concentrate on the potential that UML has with respect to reuse and iterative evolution. The lack of a precise semantics for UML is one of the main inhibitors and needs to be overcome in order to add reuse and evolution features to UML.

In *A Formal Approach to Relationships in The Unified Modeling Language* G. Övergaard presents parts of a formal specification of the Unified Modeling Language. The paper focuses on the relationship constructs in UML, such as Association, Import and different kinds of Generalization. It gives a "meta-operational" semantics as it focuses less on "what" an UML concept means, but instead on "how" an UML concept is to be manipulated.

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. This is demonstrated by modelling an SDL specification for the so called bump game.

In their paper *A Minimal Transition System Semantics for Lightweight Class- and Behavior Diagrams* R. Wieringa and J. Broersen define semantics for a subset of UML, which they call “lightweight UML”. The semantics for lightweight UML class diagrams and ultra-lightweight statecharts is given in terms of labeled step transition systems that embody a minimal change, maximal step semantics, and in which changes generated in a step have effect in the following step. In order to define the semantics, they introduce dynamic step logic.

References

[1] UML Group. Unified Modeling Language. Rational Software Corporation, Version 1.1, July 1997.