FormSERA Workshop on Formal Methods in Software Engineering Rigorous and Agile Approaches

2nd of June 2012 at ICSE'2012 in Zürich (CH)

Stefan Gruner, Bernhard Rumpe
sg@cs.up.ac.za, rumpe@se-rwth.de
DOI: 10.1145/2382756.2382777
http://doi.acm.org/10.1145/2382756.2382777

ABSTRACT

This report summarizes the activities and results of the FormSERA workshop on Formal Methods in Software Engineering – Rigorous and Agile Approaches. The workshop took place on the 2nd of June 2012 in Zürich (CH) under the umbrella of the 34th International Conference on Software Engineering, ICSE 2012, see *http://www.formsera.org/*
Keywords: Formal Methods (FM), Agile Methods (AM), Combination FM and AM

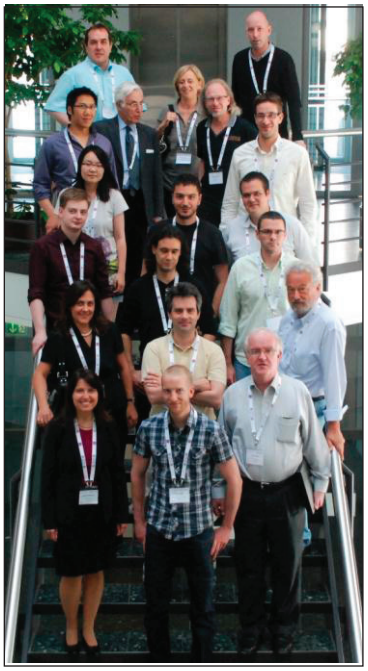
OVERVIEW

The *FormSERA'2012* workshop at ICSE'2012 was the result of an attempt to merge of two previously independent workshops with the idea of gaining synergy effects from two different, but closely related communities. The Formal Methods and Agile Methods (FM+AM'2012) community was represented by *Bernhard Rumpe* and *Stefan Gruner*.[1] *Stefania Gnesi* and *Nico Plat* –see Acknowledgments– represented the Formal Methods in Software Engineering community. This merger resulted in the new workshop called "FormSERA": *Formal Methods in Software Engineering: Rigorous and Agile Approaches*.

Our workshop FormSERA'2012 addressed the use of formal methods in software development. Formal methods differ from other software engineering techniques in that they demand and exploit a mathe-matically rigorous semantic basis for the tools and notions used. Such sound foundations permit the analysis of software engineering artefacts to a depth, and with a degree of automation, that is otherwise impossible to achieve.

Many studies have shown that formal techniques can be used in real industrial settings if knowledge and tool-support are also provided. But the maturing of formal techniques into industrial software engineering really involves providing notations and tools that are readily understood by software practitioners who are –unfortunately– often not sufficiently educated in classical computer science and theoretical informatics. Moreover, such tools must also be integrated into practical "workflows" which are beyond the simplified ideal-assumptions by which some of the earlier formal methods research was characterized. Examples include deployment of formal methods in conjunction with structured requirements analysis, software architecture and programming practices including aspect-oriented techniques and agile development. Our workshop called for original papers in all these fields.

Making progress in the industrial usability of formal methods requires also bringing together formalists and theoreticians together with soft-ware engineers from a wide range of backgrounds. In spite of the jargon and language problems which such encounters will inevitably entail, the need to achieve some dialogue between the fairly small formal methods community and the much larger community of software engineers and practitioners was our main motivation to establish our workshop under the umbrella of the ICSE conference.



Some participants of the FormSERA'2012 workshop in at the University of Zürich (Campus Irchel), Switzerland.

INVITED LECTURE

Michael Jackson (Open University and University of Newcastle, England) addressed the participants on the limits of formalization. His lecture (which was unfortunately not written down as a full paper for the workshop's proceedings), is summarized as follows:

"*Two fundamental pillars of Software Engineering practice are formalism and structure. Formalism allows engineers to reason rigorously about the system in hand; structure allows them to understand its purposes and behaviours. In the constructive activity of system development structure must therefore take precedence. The central role of formalism is to check and verify –or, where necessary, correct– the products of more informal modes of thought. In this talk these ideas are explored in the*

[1] Stefan Gruner& Bernhard Rumpe (Eds.): *FM+AM'2010 Second International Workshop on Formal Methods and Agile Methods*. Lecture Notes in Informatics, Vol. 179, September 2010, GI-Publ., ISBN 978-3-88579-273-4.

[GR12] Stefan Gruner, Bernhard Rumpe
FormSERA Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA),
FormSERA Workshop June 2nd, 2012 at ICSE 2012, Zürich,
in: ACM SIGSOFT Software Engineering Notes, vol. 37, no. 6 pages 28-30, November 2012.
www.se-rwth.de/publications

context of an illustrative system. The large structure of the system functionality is discussed, together with the nature of the components of that structure. Informal criteria of functional simplicity are presented. The inescapable mismatch between an intelligible functional structure and implementable software architecture is exposed. The role of formalism in these concerns is suggested".[2]

ACCEPTED PAPERS

After a thorough review procedure with at least three reviews per paper submitted, the following 8 papers have been chosen for presentation at FormSERA'2012. These papers are published in the IEEE Xplore Digital Library.[3] Prior to review 16 papers had been submitted, which makes an acceptance rate of 50% percent.

Further Steps towards Efficient Runtime Verification – Handling Probabilistic Cost Models

This paper by *Antonio Filieri* and *Carlo Ghezzi* is about model-checking techniques for systems with probabilistic models the probabilities of which are subject to change at runtime. The paper extends previous work of the authors for Discrete Time Markov Models (DTMC) by considering that models are enriched with rewards functions for states and transitions. The considered properties are formulas of R-PCTL, an extension of PCTL that allows to express properties concerning the cumulated reward in a path. The paper proposes a technique that reduces the online phase of model-checking to the evaluation of polynomial formulas that depend on the actual probabilities and actual reward values. The time complexity of the evaluation of these formulas is analyzed and an empirical evaluation is also conducted for one of the type of formulas that involve rewards.

Language Engineering as an Enabler for Incremental Formal Analyses

This paper by *Daniel Ratiu*, *Markus Voelter*, *Bernhard Schätz* and *Bernd Kolb* notices a semantic gap between today's general purpose programming languages on the one hand and the input languages of formal verification tools on the other hand. This makes integrating formal analyses into the daily development practice artificially complex. The authors advocate that the use of language engineering techniques can substantially improve this situation along three dimensions. First, more abstract and thus more analyzable domain specific languages can be defined, avoiding the need for abstraction recovery from programs written in general purpose languages. Second, restrictions on the use of existing languages can be imposed and thereby more analyzable code can be obtained. Third, by expressing verification conditions and the verification results at the domain level, they are easier to define and the results of analyses are easier to interpret by end users. The authors exemplify their approach with three domain specific language fragments integrated into the C programming language, together with a set of analyses: completeness and consistency of decision tables, model-checking-based analyses for a dialect of state machines and consistency of feature models. Their examples are based on the "mbeddr" stack, an extensible C language and IDE for embedded software development.

Making Sense of Recursion Patterns

This paper by *Paul Bailes* and *Leighton Brough* analyses the value of Functional Programming for contemporary Software Engineering. Recursion patterns (such as "foldr") have the potential to supplant explicit recursion in a viable sub-recursive functional style of programming. Especially however in order to be able to eschew explicit recursion entirely, even in the definition of new recursion patterns, it's essential to identify and validate a minimal set of basic recursion patterns. The immediate plausibility of "foldr" is validated by its application to the implementation of functions and recursion patterns, and especially by an abstract characterization of the programming devices used in these applications used to overcome complementary information deficiencies in data and control.

Scrum goes Formal – Agile Methods for Safety-Critical Systems

This paper by *Sune Wolff* (a former contributor to the related FM+AM workshop) states that formal methods have only low penetration in industry but have the potential for much wider use. The use of agile methods has been highly limited in development of safety critical systems due to the lack of formal evaluation techniques and rigorous planning. A combination of formal methods and agile development processes can potentially widen the use of formal methods in industry as well as enabling the use of agile methods in development of safety-critical systems. Wolff's paper describes a way to add the use of formal methods to the agile development process Scrum. Experiences from using a variant of the strategy in an industrial case are described.

Revisiting Modal Interface Automata

This paper by *Ivo Krka* and *Nenad Medvidovic* states that modern software systems are typically built of components that communicate through their external interfaces. A component's behavior can be effectively described using finite state automata-based formalisms. The basic formalism, labeled transition systems, describes the behavior of a component in terms of states and labeled transitions. More advanced formalisms, such as modal transition systems and interface automata, extend LTS to incorporate additional information related to interface operation controllability and the possible partiality of a component's specification. Capturing controllability and partiality aspects of a component's specification facilitates checking interface compatibility, checking whether one component can safely replace another component, and checking whether one specification is a proper refinement of another specification. For their paper the authors studied the existing definitions of these three types of checks and exemplified their limitations in the context of the richest class of component behavior specifications, modal interface automata (MIA). The authors also outline a set of enhancements to MIA as possible solutions to those limitations.

Automated Continuous Quality Assurance

This paper by *Johannes Neubauer*, *Bernhard Steffen*, *Oliver Bauer*, *Stephan Windmüller*, *Maik Merten*, *Tiziana Margaria* and *Falk Howar*, presents a case study which illustrates the power of active learning for enabling automated quality assurance of evolving systems. It is shown how the development of OCS, Springer's online conference system, is accompanied by

[2] *http://www.formsera.org/FormSERA/Program.html*
[3] *http://ieeexplore.ieee.org/*

continuous learning-based testing, that, by its nature, maintains the synchrony of the running application and the learned (test) model. The evolution of the test model clearly indicates which portions of the system remain stable and which are altered. This approach comprises classical regression testing and feature interaction detection.

EMF to CSP – A Tool for the Lightweight Verification of EMF

This paper by *Carlos González*, *Fabian Büttner*, *Robert Clarisó* and *Jordi Cabot*, states that the increasing popularity of MDE results in the creation of larger models and model transformations. Hence converting the specification of MDE artefacts is an error-prone task. Therefore mechanisms to ensure quality and absence of errors in models are needed to assure the reliability of the MDE-based development process. Formal methods have proven their worth in the verification of software and hardware systems. However the adoption of formal methods as a valid alternative to ensure model correctness is compromised by the inner complexity of the task. To circumvent this complexity it is common to impose limitations, such as reducing the type of constructs that can appear in the model, or turning the verification process from automatic into user-assisted. Considering such limitations as counter-productive for the adoption of formal methods, the authors present *EMFtoCSP*, a new tool for the fully automatic, decidable and expressive verification of EMF models that uses constraint logic programming as the underlying formalism.

Augmenting Event-B Modelling with Real-Time Verification

This paper by *Alexei Iliasov*, *Linas Laibinis*, *Elena Troubitsyna*, *Alexander Romanovsky* and *Timo Latvala* states that a large number of dependable embedded systems have stringent real-time requirements imposed on them. An analysis of real-time behaviour is usually conducted at implementation-level. However, it is desirable to obtain an evaluation of real-time properties early at the development cycle, i.e., at the modelling stage. In their paper the authors present an approach to augmenting Event-B modelling with verification of real-time properties in Uppaal. They show how to extract a process-based view from an Event-B model that together with introducing time constraints allows them to obtain a timed automata model – the input model of Uppaal. The authors illustrate their approach by the development and veri-fication of the data-processing on-board software of the *BepiColombo* Mission.

PROGRAMME COMMITTEE, REVIEWERS

Yamine Ait-Ameur (IRIT/ENSEEIHT France), *Gogul Balakrishnan* (NEC Labs USA), *Manfred Broy* (TechnischeUniversitätMünchen Germany), *Marsha Chechik* (University of Toronto Canada), *Jürgen Dingel* (Queens University Canada), *Patrick Heymans* (University of Namur Belgium), *Alessandro Fantechi* (University of Firenze Italy), *Mike Hinchey* (LERO Ireland), *Axel van Lamsweerde* (University of Louvain Belgium), *Peter Gorm Larsen* (Engineering College of Aarhus Denmark), *Thierry Lecomte* (ClearSy France), *Shaoying Liu* (Hosei University Japan), *Antonia Lopes* (University of Lisboa Portugal), *Michael Löwe* (Fachhochschule für die Wirtschaft Hannover Germany), *Tiziana Margaria* (Universität Potsdam Germany), *Steve Riddle* (Uni-versity of Newcastle England), *Matteo Rossi* (Politecnico di Milano Italy), *Bernhard Schätz* (Fortiss Germany), *Wolfram Schulte* (Microsoft USA), *Chandrasekaran Subramaniam* (Rajalakshmi Engineering Col-lege Chennai India), *Elena Troubitsyna* (Abo University Finland), *Sebastian Uchitel* (Universidad de Buenos Aires Argentina), *Willem Visser* (Universiteit van Stellenbosch South Africa), *Bartosz Walter* (University of Technology Poznan Poland), *Fatiha Zaïdi* (LRI/CNRS France).

ACKNOWLEDGMENTS

Many thanks to *Nico Plat* (West Consulting BV – The Netherlands) as well as to *Stefania Gnesi* (ISTI CNR – Italy) for their cooperation in organizing and conducting the workshop. We are also thankful to our invidted speaker *Michael Jackson* for presenting the above-mentioned keynote lecture. Thanks go to the organization FME for having sponsored Michael Jackson's keynote talk. Thanks also to all reviewers and PC members for their diligence and scrutiny. Last but not least thanks to the entire ICSE'2012 team, particularly *Martin Glinz* in Zürich, for having accepted this workshop under the umbrella of ICSE and for having provided all the necessary local support.