Softw Syst Model (2015) 14:3–4
DOI 10.1007/s10270-014-0404-6

GUEST EDITORIAL

SEFM: software engineering and formal methods

Gilles Barthe · Alberto Pardo · Gerardo Schneider

Received: 3 February 2014 / Accepted: 9 February 2014 / Published online: 22 February 2014

1 Introduction

Software engineering is concerned with the design, development and maintenance of large software using systematic techniques. Research performed in software engineering includes a variety of topics like studies on psychological aspects of teams developing software, new agile methodologies, software processes and architectures, model-based development approaches, and validation and verification, to mention only a few. Though the discipline has considerably advanced in the last twenty years, it still lacks a formal and rigorous approach in some of its specific topics or subareas, making it difficult to assess in general that a software product satisfies its requirements. Besides, some specific software development standards require a certain minimum level of rigorousness in the development and validation of tools in order to be certified, justifying the use of more formal techniques.

Formal methods consist of mathematically based techniques for the specification and verification of software (and hardware) systems. The main motivation for developing and using formal methods is the understanding that without well-founded and rigorous languages, techniques and algorithms it is in general difficult to provide evidence that the software does what it is supposed to do, with a minimum level of confidence in its quality. Formal methods can play different roles in the software development process. It can help to make system and software descriptions more precise and unambiguous, and provide support for analysis, in some cases automatically. As the application of formal methods in real industrial software systems is only feasible when supported by tools, much of the effort of researchers in the area today is put not only on the development of different specification languages, verification algorithms and techniques, but also on the development of reliable and usable tools.

The aim of this special issue is to provide a collection of papers on different aspects of formal methods, hoping that advances in this area will help their applicability in software engineering.

G. Barthe (✉)
IMDEA Software Institute, Madrid, Spain
e-mail: gilles.barthe@imdea.org
URL: http://software.imdea.org/∼gbarthe

A. Pardo
Instituto de Computación, Facultad de Ingeniería,
Universidad de la República, Montevideo, Uruguay
e-mail: pardo@fing.edu.uy
URL: http://www.fing.edu.uy/∼pardo

G. Schneider
Department of Computer Science and Engineering,
University of Gothenburg, Gothenburg, Sweden
e-mail: gerardo@cse.gu.se
URL: http://www.cse.chalmers.se/∼gersch

2 Scope and selection of papers

This volume is a special issue containing a selection of papers presented at the 9th International Conference on Software Engineering and Formal Methods (SEFM) held on November 14–18, 2011, in Montevideo, Uruguay. The aim of SEFM is to bring together practitioners and researchers from academia, industry and government, to advance the state of the art in formal methods, to scale up their application in software industry, and to encourage their integration with practical engineering methods.

Among those accepted at the conference, we have invited a selection of papers for further submission to this special issue, and among those submitted we have finally accepted the following 10 articles.

In the first paper of this volume, *Translating between Alloy specifications and UML Class Diagrams annotated with OCL*, Alcino Cunha, Ana Garis, and Daniel Riesco present a model transformation from alloy to UML class diagrams annotated with OCL (UML+OCL). The proposed transformation enables a smooth integration of alloy in the current model-driven engineering contexts, by allowing UML+OCL specifications to be transformed to Alloy for validation and verification.

The next paper, *Verification of B+ Trees by Integration of Shape Analysis and Interactive Theorem Proving*, by Gidon Ernst, Gerhard Schellhorn, and Wolfgang Reif, proposes an approach integrating shape analysis and interactive theorem proving for proving correctness of pointer-manipulating programs. The approach uses shape analysis to automatically discharge proof obligations for various data structure properties.

In *TacoFlow: Optimizing SAT Program Verification Using Dataflow Analysis*, Diego Garbervetsky, Bruno Cuervo Parrino, Juan Galeotti, and Marcelo Frias present TacoFlow, a tool to translate programs annotated with contracts to a SAT problem which is then solved resorting to off-the-shelf SAT solvers. It uses dataflow analysis in order to discard propositional variables that describe intermediate program states. It also presents an empirical evaluation on the effect of removing those variables at different levels of abstraction.

Jeremy Morse, Lucas Cordeiro, Denis Nicole, and Bernd Fischer, present in *Model Checking LTL Properties over ANSI-C Programs with Bounded Traces* an approach to extend context-bounded software model checking to safety and liveness properties expressed in linear time temporal logic (LTL), by checking the actual C program rather than an extracted abstract model. An extended, four-valued LTL semantics is used to handle the finite traces that bounded model checking explores. The approach is demonstrated on the analysis of the sequential firmware of a medical device and a small multi-threaded control application.

In *Procedure-Modular Specification and Verification of Temporal Safety Properties*, Siavash Soleimanifard, Dilian Gurov, and Marieke Huisman describe ProMoVer, a tool for fully automated procedure-modular verification of Java programs equipped with method-local and global assertions that specify safety properties of sequences of method invocations. The tool is evaluated on Java card and web-based applications.

Daniel Delahaye, Mélanie Jacquel, Karim Berkani, and Catherine Dubois present in *Verifying B Proof Rules using Deep Embedding and Automated Theorem Proving* a formal and mechanized framework for verifying proof rules of the B method. The framework contains a set of tools relying on a deep embedding of the B theory within the logic of the Coq proof assistant. This toolkit allows to automatically generate the required properties to be checked for given proof rules.

In *Improving the SAT Modulo ODE Approach to Hybrid Systems Analysis by Combining Different Enclosure Methods*, Andreas Eggers, Nacim Ramdani, Nedialko Nedialkov, and Martin Fränzle present a novel combination of enclosure methods for ordinary differential equations with the iSAT solver for large Boolean combinations of arithmetic constraints, in order to automatically verify hybrid discrete-continuous systems. Experiments are conducted on classic benchmarks as well as on a conveyor belt system consisting of parallel components, a slip-stick friction model with nonlinear dynamics and flow invariants and several dimensions of parameterization.

In *Synchrony and Asynchrony in Conformance Testing*, Neda Noroozi, Ramtin Khosravi, Mohammad Mousavi, and Tim Willemse present and compare different notions of conformance testing based on labeled transition systems. Several theorems are proven concerning when synchronous test cases are sufficient for checking all aspects of conformance that are observable by asynchronous interaction with the implementation under test.

The paper *Runtime Verification of Component-Based Systems in the Behavior, Interaction and Priority (BIP) Framework with Formally-Proved Sound and Complete Instrumentation* by Yliès Falcone, Mohamad Jaber, Thanh-Hung Nguyen, Marius Bozga, and Saddek Bensalem presents a method to integrate runtime verification into the BIP component-based framework for heterogeneous systems. The method has been implemented in RV-BIP, a prototype tool used to validate the whole approach on a robotic application.

In the last paper, *Broadcast Psi-calculi with an Application to Wireless Protocols*, Björn Victor, Johannes Borgström, Shuqin Huang, Magnus Johansson, Palle Raabjerg, Johannes Åman Pohjola, and Joachim Parrow extend Psi-calculi with primitives for broadcast communication in order to model wireless protocols, showing that the additions preserve the purity of the Psi-calculi semantics. Formal proofs of the standard congruence and structural properties of bisimilarity are given. A model of the wireless ad hoc routing protocol LUNAR is presented, and a basic reachability property is verified.

We would like to thank the organizing committees of SEFM 2011 for setting up such a successful event, and the programme committee and reviewers of the conference and the special issue for helping with the selection of papers. Finally, we would like to address our special thanks to Martin Schindler and the editorial team of SoSyM for their continuous support and encouragement during the edition of this issue.