International Journal on Software Tools for Technology Transfer (2021) 23:285–287
https://doi.org/10.1007/s10009-020-00589-6

GENERAL

Special Issue: FASE 2019

Automated model analysis tools and techniques presented at FASE 2019

Reiner Hähnle[1] · Wil van der Aalst[2]

Published online: 7 September 2020

Abstract
This special issue contains substantially revised and extended versions of some of the best papers presented at the 22nd International Conference on Fundamental Approaches to Software Engineering in 2019. All papers share the common theme that they are either concerned with model-based analysis of systems or they develop methods in its service.

Keywords Model transformation · Test generation · Model synchronization · Graph repair · Data flow analysis

1 Introduction

Two technology-driven developments are currently having a massive impact on the way we live: the digitalization of everything, and world-wide, wireless connectivity over standardized protocols. The consequence is that all kinds of processes and services move from the physical realm to the software domain. In this sense, software is becoming quickly—and quite literally—the fabric of our technological and societal infrastructure. For this reason, the availability of reliable, secure, and trusted software becomes ever more crucial.

At the same time, expectations are sometimes unrealistic. When building a bridge one would never dream of making drastic changes, e.g., adding two more lanes and a bus stop. However, when it comes to software we expect that things can be changed at any point in time. Also software is used in environments that did not exist during design. This is like designing a bridge for a canal in Amsterdam and deploying copies of it in Alaska and the Sahara. Subsequently, these deployed copies are updated every two weeks.

Solid software engineering foundations and a good understanding of the fundamental properties of software are essential for building reliable and sustainable "software bridges". This requires substantial advances in software science. And here the academics and practitioners involved with FASE continue to make highly relevant contributions.

Here we showcase a sample of these contributions. Originating from the 2019 edition of the International Conference on Fundamental Approaches to Software Engineering[1] (FASE), this special issue contains revised and substantially extended versions of the strongest papers.

FASE is concerned with the foundations on which software engineering is built. Accepted papers are supposed to contribute novel contributions to making software engineering a more mature and soundly based discipline. They must be supported by appropriate arguments and validation. Contributions combining the development of conceptual and methodological advances with their formal foundations and tool support are particularly encouraged. Specific topics of interest include:

- Software engineering as an engineering discipline, including its interaction with and impact on society and economics;
- Requirements engineering: capture, consistency, and change management of software requirements;
- Software architectures: description and analysis of the architecture of individual systems or classes of applications;

✉ Reiner Hähnle
reiner.haehnle@tu-darmstadt.de

Wil van der Aalst
wvdaalst@pads.rwth-aachen.de

[1] Technical University of Darmstadt, Darmstadt, Germany

[2] RWTH Aachen University, Aachen, Germany

[1] Held as part of the European Joint Conference on Theory and Practice of Software (ETAPS) in Prague, Czech Republic, 8–11 April 2019.

- Specification, design, and implementation of particular classes of systems: (self-)adaptive, collaborative, embedded, distributed, mobile, pervasive, cyber-physical or service-oriented applications;
- Software quality: (static or run-time) validation and verification of functional as well as non-functional software properties using theorem proving, model checking, testing, analysis, simulation, refinement methods, metrics or visualization techniques;
- Model-driven development and model transformation: meta-modeling, design and semantics of domain-specific languages, consistency and transformation of models, generative architectures;
- Software processes: support for iterative, agile, and open source development;
- Software evolution: refactoring, reverse and re-engineering, configuration management, also architectural change, or aspect-orientation.

2 Contributions

We received 94 abstract submissions of which 74 were turned into full submissions (63 research papers, 5 tool papers, and 6 demo papers). We had submissions from 33 countries. Of the 74 actually submitted papers, 24 papers were accepted after thorough reviewing and discussions among the Program Committee members (20 research papers, 2 tool papers, and 2 demo papers). Next to the 30 PC members, there were 100 external reviewers. For the fourth time, FASE used a double-blind reviewing process.

The guest editors of this special issue invited six of the top-ranked papers in the peer-review process to submit an extended version. The invitations were not just based on score, but also on significance. Only in hindsight, though, the editors noted to their surprise that all of the selected papers share a common thread:

Today's software is so complex that it is often advisable not to work with the actual code, but with a *model* that contains only its essential aspects and abstracts away from artifacts that are irrelevant for the performed analysis. All papers in this special issue have in common that they work at the level of models, even though the considered models are, of course, differing and are chosen to suit the intended analysis or application domain. It is also noteworthy that four out of six papers feature an *incremental* analysis approach as one of their main contributions. Again, this is no coincidence: in the absence of strong decomposition principles, incremental computation is often the only way to handle large or complex models. Finally, it needs to be stressed that *all* of the included papers come with an implementation of the theoretical advances they purport and feature an experimental evaluation of some kind. Several papers even demonstrate their findings on realistic, industrial systems. This not only reflects the growing maturity of the field of Fundamental Software Aspects, but also the spirit of the STTT journal.

The six invited papers are as follows:

- *Incremental Execution of Rule-based Model Transformation* by A. Boronat [2] presents the design of a model change propagation mechanism for executing change-driven model transformations, which has been implemented in YAMTL. Change propagation mechanisms form the foundation for maintaining consistency between large models. The novelty of the approach lies in the standardized representation of model changes. The VIATRA CPS benchmark has been used to evaluate the work.
- *Cooperative, Verifier-Based Testing with CoVeriTest* by Beyer & Jakobs [1] is a test generation approach based on verification attempts: from a solved reachability problem, one can generate a test case. This idea is not new, the innovation lies in how the approach is made *feasible*: the reachability analysis is configurable with different types of information and levels of abstraction. These ideas are implemented in the CoVeriTest tool.
- *Avoiding Unnecessary Information Loss* by Fritsche et al. [4] focuses on Triple Graph Grammars (TGGs) for model synchronization, i.e., the task of restoring the consistency between two models after model changes. Model synchronization poses challenges, e.g., one should be able to synchronize changes without unnecessary loss of information and to show a reasonable performance. The novel idea to address these challenges is to add shortcut rules, a special form of generalized TGG rules that allow taking back one edit action and to perform an alternative one. The authors show that repair rules derived from shortcut rules allow for incremental model synchronization with considerably decreased information loss and improved runtime compared to existing approaches.
- *A Logic-Based Incremental Approach to Graph Repair Featuring Delta Preservation* by Schneider et al. [5] tackle the problem of graph repair, i.e., to find a minimal set of changes that make two given type graphs consistent. This an essential capability of tools in model-based design, because graphs are frequently used as abstract representations of code. Numerous graph repair approaches are known. This one distinguishes itself by being incremental and founded on a formal logic as its semantics. The algorithms are implemented in the tool AutoGraph.
- *Formal Testing of Timed Graph Transformation Systems using Metric Temporal Graph Logic* by Schneider et al. [6] is about a model of embedded real-time systems called timed graph transformation systems (TGTS). During model-based development of such systems it is

essential that one can test the model, but, while TGTS are adequate for modeling, they are too expressive to directly permit testing. The authors define a series of transformations that reduce testing of TGTS to satisfiability checking in a metric temporal graph logic. All this is implemented in the AutoGraph system.

– *PolyGraph: A Data Flow Model with Frequency Arithmetic* by Dubrulle et al. [3] proposes an extension to static data flow paradigms that includes frequency constraints and adjustable communication rates. PolyGraph, the proposed data flow formalism, extends SDF with synchronous firing semantics for the actors. The authors define a novel algorithm for checking liveness of a given polygraph. The proposed framework was evaluated using both small (but realistic) examples as well as a large set of automatically generated models.

Acknowledgements We are grateful to the editors of the *Intl. J. on Software Tools for Technology Transfer* for supporting this special issue. We would also like to thank all the Program Committee members of FASE 2019 for their thorough and professional work. A big thank you goes to the reviewers involved in this special issue: Several of the papers have a substantial size and contain complex technical developments. Still, you never let us down and came back with extensive constructive suggestions. Your work is deeply appreciated. Finally, we thank all the authors for creating substantially extended and revised versions of their conference papers.

Funding Open Access funding provided by Projekt DEAL.

References

1. Beyer, D., Jakobs, M.C.: Cooperative, verifier-based testing with CoVeriTest. Int. J. Softw. Tools Technol, Transfer (in this issue) (2020)
2. Boronat, A.: Incremental execution of rule-based model transformation. Int. J. Softw. Tools Technol, Transfer (in this issue) (2020)
3. Dubrulle, P., Kosmatov, N., Gaston, C., Lapitre, A.: PolyGraph: a data flow model with frequency arithmetic. Int. J. Softw. Tools Technol, Transfer (in this issue) (2020)
4. Fritsche, L., Kosiol, J., Schürr, A., Taentzer, G.: Avoiding unnecessary information loss. Int. J. Softw. Tools Technol, Transfer (in this issue) (2020)
5. Schneider, S., Lambers, L., Orejas, F.: A logic-based incremental approach to graph repair featuring delta preservation. Int. J. Softw. Tools Technol, Transfer (in this issue) (2020)
6. Schneider, S., Maximova, M., Sakizloglou, L., Giese, H.: Formal testing of timed graph transformation systems using metric temporal graph logic. Int. J. Softw. Tools Technol, Transfer (in this issue) (2020)