Formal methods and automated verification of critical systems

Maurice H. ter Beek[1] · Stefania Gnesi[1] · Alexander Knapp[2]

Abstract

Critical (software) systems are all around us. These systems are typically characterised by stringent dependability requirements and demand elevated levels of robustness and fault tolerance. To assure that they function as intended and provide a number of quality guarantees, formal methods and automated verification techniques and tools have been in use in the engineering of such critical systems for many years now. In this introduction to the special issue FMICS–AVoCS on "Formal Methods and Automated Verification of Critical Systems", we outline a number of recent achievements concerning the use of formal methods and automated verification techniques and tools for the specification and analysis of critical systems from a variety of application domains. These achievements are represented by six selected papers: five were selected from the joint 21st *International Workshop on Formal Methods for Industrial Critical Systems* and 16th *International Workshop on Automated Verification of Critical Systems* (FMICS–AVoCS 2016), while one of them was selected after an open call for papers.

Keywords Formal methods · Automated verification · Critical systems

1 Introduction

We are surrounded by a fair amount of critical (software) systems with stringent dependability requirements and necessitating elevated levels of robustness and fault tolerance. It is of paramount importance that such systems (e.g. safety-critical, business-critical, performance-critical) function as intended and provide a number of overall quality guarantees. Thus, the main causes of software failures—such as requirements defects, design faults and incorrect implementations—need to be excluded with the highest levels of assurance. To this aim, formal methods and automated verification techniques and tools have been in use in the engineering of such critical (software) systems for many years now [1,2] and their use is currently a hot topic in numerous application domains [3–10].

Formal methods are specification languages for describing the behaviour of a (software) system as a model with a precise semantics, thus allowing their associated formal verification tools to perform analyses over these system models [11]. Similar to other engineering disciplines, the envisioned advantage of their use is the expectation that appropriate mathematical modelling and analysis can contribute to the correctness of the developed systems by eliminating flaws during the initial (software) development phases, i.e. well before implementation, and by ensuring robust and fault-tolerant systems that perform as specified even in uncertain or inconsistent environments.

This special issue dedicated to "Formal Methods and Automated Verification of Critical Systems" contains a total of six papers. Five of them are extended versions of selected papers from the joint 21st *International Workshop on Formal Methods for Industrial Critical Systems* and 16th *International Workshop on Automated Verification of Critical Systems* (FMICS–AVoCS 2016), while one of them was selected after an open call for papers.

The 21st *International Workshop on Formal Methods for Industrial Critical Systems* and the 16th *International Workshop on Automated Verification of Critical Systems* (FMICS–AVoCS 2016), which were organised as a joint event from 26 September to 28 September 2016, in Pisa, Italy,

✉ Maurice H. ter Beek
terbeek@isti.cnr.it

Stefania Gnesi
gnesi@isti.cnr.it

Alexander Knapp
knapp@informatik.uni-augsburg.de

[1] Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy

[2] Institute for Software and Systems Engineering, Universität Augsburg, Augsburg, Germany

called for contributions on the following, non-exhaustive, topics of interest:

- Design, specification, refinement, code generation and testing of critical systems based on formal methods.
- Methods, techniques and tools to support the automated analysis, certification, debugging, learning, optimisation and transformation of critical systems, in particular distributed, real-time systems and embedded systems.
- Automated verification (e.g. model checking, theorem proving, SAT/SMT constraint solving, abstract interpretation) of critical systems.
- Verification and validation methods addressing shortcomings of existing methods with respect to their industrial applicability (e.g. scalability and usability issues).
- Tools for the development of formal design descriptions.
- Case studies and experience reports on industrial applications of formal methods, focussing on lessons learnt or on the identification of new research directions.
- Impact of the adoption of formal methods on the development process and associated costs.
- Application of formal methods in standardisation and industrial forums.

The proceedings of FMICS–AVoCS 2016 have been published in Springer's Lecture Notes in Computer Science series [12].

Shortly after the joint event, in November 2016, an open call for papers devoted to "Formal Methods and Automated Verification of Critical Systems" was issued. According to the call, research papers containing novel, previously unpublished results in all areas related to the topics of the FMICS and AVoCS workshop series were sought for. This resulted in the submissions of 14 papers. Based upon a thorough reviewing process, the editors decided to accept nine papers; three of them appeared in a dedicated special issue on "Formal Methods for Transport Systems" of Springer's International Journal on *Software Tools for Technology Transfer* (STTT) [10], due to their explicit focus on transport systems, while six papers are included in this special issue.

In the remainder of this introduction, we will briefly present and contextualise the contributions of the papers contained in this special issue, followed by a discussion of the overall impact of this special issue.

2 Selected papers

The first paper of this special issue, *Qualitative and quantitative analysis of safety-critical systems with S#* by Leupolz et al. [13], presents an integrated, uniform framework for simulating and verifying safety-critical systems. This framework S# is directly based on the C# programming language and development tools, for user-friendliness, practicality, and versatility. For safety-critical systems, dedicated support for fault modelling is added. Both qualitative analysis, based on linear temporal logic, and quantitative analysis using fault probabilities can be conducted. The S# approach utilises the LTSmin tool [14] for state space generation, feeding LTSmin with state data directly computed from the C# program. On the one hand, LTSmin can then be used for explicit-state model checking for qualitative analysis, like direct-cause-consequence analysis (DCCA). On the other hand, a Markov chain can be derived for qualitative analysis when the faults are labelled with probabilities. The effectiveness and the efficiency of the approach are demonstrated on several case studies.

The second paper of this special issue, *Runtime verification of autopilot systems using a fragment of MTL-$\int$* by de Matos Pedro et al. [15], introduces a novel approach to runtime verification of hard real-time systems involving explicit time and duration. The approach uses the three-valued restricted metric temporal logic with durations RMTL-$\int_3$. As "bare-metal" real-time embedded systems running on X86 and ARM processors are targeted, monitor synthesis delivers C++11 code. A hierarchy of monitors is synthesised that provide hard timing guarantees on each level. The expressiveness of the logic and the synthesis approach is first discussed on two uses cases: the first demonstrates an application to resource models exhibiting under- and over-loading conditions for task. The second use case evaluates the likelihood of tasks remaining unscheduled based on the overload of another task. This use case is feasible as also conditional probabilities can be represented in the logic and the synthesised monitors. The overall framework is then successfully evaluated on the PixHawk platform for an autopilot system showing its applicability in practice.

The third paper of this special issue, *High-level frameworks for the specification and verification of scheduling problems* by Chadli et al. [16], contributes to the highly interesting and timely issue of schedulability of Cyber-Physical Systems. Leveraging on a model-based rather than analytical approach, Chadli et al. present so-called "simplicity-driven" high-level specification and verification frameworks, based on variants of (probabilistic) timed automata and supported by the well-known Uppaal (SMC) toolset [17], to describe and analyse scheduling problems (in particular correctness, optimisation and monitoring). These frameworks can moreover be constructed by a domain-specific tool generator through meta-modelling, thus facilitating the task of designing systems. A convincing feature is the adoption of a user-friendly (graphical) approach: the high-level specifications are automatically translated to formal models (thus hiding the latter from practitioners) to enable for the formal verification, and the analysis results are transformed back to the high-level specification language to facilitate their interpretation (graphically). The effectiveness of the proposed approach is showcased by experimenting with high-level frameworks for two scheduling case studies.

In the fourth paper of this special issue, *Integrated formal verification of safety-critical software* by Ge et al. [18], a formal verification process based on the commercial Systerel Smart Solver (S3) toolset[1] [19] for the development of safety-critical embedded software systems is presented. The reader is guided through this process by means of an Automatic Rover Protection system implemented onboard a robot, which is a fitting example of a non-trivial safety-critical embedded system, with distributed components, some amount of central control and some amount of independence. Furthermore, the paper offers a solution to the use of floating-point arithmetics on bit level, which—in spite of some scalability issues—nicely shows the limits of what is feasible with current technology. This paper is of particular interest to practitioners in the field of safety-critical systems, as Ge et al. provide detailed guidance on how to apply which techniques under what circumstances, and they do so from system requirements down to code.

The fifth paper of this special issue, *Model-based testing strategies and their (in)dependence on syntactic model representations* by Huang and Peleska [20], presents a fresh look at model-based testing, shifting the focus from the apparent syntactic representation to the true semantic contents of models. In fact, classical model-based testing relies on the concrete syntactical form of a model for test case generation. This may lead to test suites of different strength for different syntactical models which, however, have the same semantics. The problem particularly arises when higher-level modelling languages like UML are used. The new approach for test case generation just relies on the language of the model described, i.e., its semantics. In particular, it is shown how to lift the traditional Wp-method for conformance testing to the language setting.

The sixth and final paper of this special issue, *Assessing SMT and CLP approaches for workflow nets verification* by Bride et al. [21], assesses the verification of workflow nets against specifications expressed in a modal logic for two symbolic approaches: one is based on Satisfiability Modulo Theories (SMT), and the other is based on Constraint Logic Programming (CLP). Workflow nets are a particular class of Petri nets suitable for describing the behaviour of business processes and workflows, and they are effectively used in industry. Therefore, the focus of the assessment is on the efficiency and scalability of the two resolution methods, which is highly relevant for technology transfer. In fact, the case studies used in the paper stem from industrial workflows obtained through collaborations with industrial partners.

The extensive experimental assessment of the two methods is performed over a huge set of several thousands of workflow nets. Two solvers are used as analysis back-ends. These are Z3 [22] (based on SMT) and SICStus Prolog[2] [23] (based on CLP). Their respective performance is thoroughly analysed and compared using four classes of workflow nets of increasing expressiveness (namely state machines, marked graphs, free-choice and ordinary workflow nets), and all combinations of valid/invalid may/must modal specifications. Based on this assessment, Bride et al. propose concrete verification strategies for modal specifications of workflow nets based on the specific features of the nets under verification and the modal specification to be verified. Their results empirically demonstrate that these methods are efficient and scalable to workflow nets consisting of up to 1000 nodes.

3 Discussion

We have briefly presented the six selected papers that constitute this special issue. The topics discussed in these papers cover a broad range of formal methods and automated verification techniques and tools, ranging from the formal verification of concrete systems like the Automatic Rover Protection system, through the modelling of safety-critical systems—including probabilistic fault modelling in the high-level programming language C# and runtime verification of embedded systems with hard timing requirements based on RMTL-$\int_3$—to semantic issues in model-based testing.

As could be expected from a special issue born out of the FMICS–AVoCS workshop series, the focus is on industry-relevant case studies, targeting practical problems related to scheduling, real-time or workflows, and on automated techniques for their modelling and verification. Consequently, oftentimes the formal details are hidden from the users, by employing high-level modelling languages and by applying automated tools that are complemented with usage guidance for practitioners. We also observe a trend in the use of real-time and probabilistic modelling. On the other hand, practical support for testing safety-critical systems remains an issue, in particular when it comes to real-time properties.

Acknowledgements We would like to thank all authors for their contributions and the reviewers of FMICS–AVoCS 2016 and in particular those of this special issue for their reviews.

References

1. Woodcock, J., Larsen, P.G., Bicarregui, J., Fitzgerald, J.S.: Formal methods: Practice and experience. ACM Comput. Surv. **41**(4), 19:1–19:36 (2009)

[1] S3 is developed, maintained and distributed by Systerel.

[2] http://sicstus.sics.se/

2. Gnesi, Stefania, Margaria, Tiziana (eds.): Formal Methods for Industrial Critical Systems: A Survey of Applications. John Wiley & Sons Inc, Hoboken (2013)
3. ter Beek, M.H., Clarke, D., Schaefer, I.: Editorial preface for the JLAMP special issue on formal methods for software product line engineering. J. Log. Algebraic Methods Programm. **85**(1), 123–124 (2016)
4. ter Beek, M.H., Lisitsa, A., Nemytykh, A.P., Ravara, A.: Automated verification of programs and Web systems. J. Log. Algebraic Methods Programm. **85**(5), 653–654 (2016)
5. ter Beek, M.H., Lafuente, A.L.: Automated specification and verification of Web-based applications. J. Log. Algebraic Methods Program. **87**, 51 (2017)
6. Güdemann, M., Núñez, M.: Preface of the special issue on formal methods in industrial critical systems. Int. J. Softw. Tools Technol. Transf. **19**(4), 391–393 (2017)
7. Ozay, N., Tabuada, P.: Guest editorial: special issue on formal methods in control. Discrete Event Dyn. Syst. **27**(2), 205–208 (2017)
8. Grov, G., Ireland, A.: Preface of the special issue on automated verification of critical systems (AVoCS 2015). Sci. Comput. Program. **148**, 1–2 (2017)
9. ter Beek, M.H., Loreti, M.: Guest editorial for the special issue on formal methods for the quantitative evaluation of collective adaptive systems (FORECAST). ACM Trans. Model. Comput. Simul. **28**(2), 81–84 (2018)
10. ter Beek, M.H., Gnesi, S., Knapp, A.: Formal methods for transport systems. Int. J. Softw. Tools Technol. Transf. **20**(3), 237–241 (2018)
11. Almeida, J.B., Frade, M.J., Pinto, J.S., de Sousa, S.M.: An overview of formal methods tools and techniques. In: Rigorous Software Development. Undergraduate Topics in Computer Science, pp. 15–44. Springer, London (2011)
12. ter Beek, M.H., Gnesi, S., Knapp, A. (eds.): Critical systems: formal methods and automated verification. In: Proceedings of the Joint 21st International Workshop on Formal Methods for Industrial Critical Systems and 16th International Workshop on Automated Verification of Critical Systems (FMICS-AVoCS 2016), volume 9933 of Lecture Notes in Computer Science. Springer (2016)
13. Leupolz, J., Knapp, A., Habermaier, A., Reif, W.: Qualitative and quantitative analysis of safety-critical systems with S#. Int. J. Softw. Tools Technol. Transf. (2018). In this issue
14. Kant, G., Laarman, A., Meijer, J., van de Pol, J., Blom, S., van Dijk, T.: LTSmin: High-performance language-independent model checking. In: Baier, C, Tinelli, C (eds.) Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2015), volume 9035 of Lecture Notes in Computer Science, pp. 692–707. Springer (2015)
15. de Pedro, M.A.., Sousa Pinto, J., Pereira, D., Pinho, L.M.: Runtime verification of autopilot systems using a fragment of MTL-$\int$. Int. J. Softw. Tools Technol. Transf. (2018). In this issue
16. Chadli, M., Kim, J.H., Larsen, K.G., Legay, A., Naujokat, S., Steffen, B., Traonouez, L.-M.: High-level frameworks for the specification and verification of scheduling problems. Int. J. Softw. Tools Technol. Transfer (2018). In this issue
17. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: Uppaal SMC tutorial. Int. J. Softw. Tools Technol. Transf. **17**(4), 397–415 (2015)
18. Ge, N., Jenn, E., Breton, N., Fonteneau, Y.: Integrated formal verification of safety-critical software. Int. J. Softw. Tools Technol. Transf. (2018). In this issue
19. Clabaut, M., Ge, N., Breton, N., Jenn, E., Delmas, R., Fonteneau, Y.: Industrial grade model checking—use cases, constraints, tools and applications. In: Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS^2 2016), pp. 85–92 (2016).
20. Huang, W.L., Peleska, J.: Model-based testing strategies and their (in)dependence on syntactic model representations. Int. J. Softw. Tools Technol. Transf. (2018). In this issue
21. Bride, H., Kouchnarenko, O., Peureux, F., Voiron, G.: Assessing SMT and CLP approaches for workflow nets verification. Int. J. Softw. Tools Technol. Transf. (2018). In this issue
22. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008), volume 4963 of Lecture Notes in Computer Science, pp. 337–340. Springer (2008)
23. Carlsson, M., Mildner, P.: SICStus prolog-The first 25 years. Theory Pract. Logic Program. **12**(1–2), 35–66 (2012)