Software Quality Journal (2020) 28:693–694
https://doi.org/10.1007/s11219-020-09508-z

Preface to the special section on improving software quality through formal methods

Yliès Falcone[1] · Leonardo Mariani[2]

Published online: 17 February 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

1 Introducing the special section

The special section is devoted to the use of formal methods for improving the quality of software systems. The included articles focus on the use of runtime techniques, such as runtime verification and trace analysis (see Falcone et al. 2013; Bartocci and Falcone 2018; Bartocci et al. 2019). Following an open call, this section resulted in 3 papers that were selected out of 8 papers after a careful and rigorous reviewing process. The papers are briefly discussed as follows.

In "Verifying Temporal Specifications of Java Programs," Spegni et al. combine satisfiability modulo theory and static analysis to approximate the behavior of Java threads. The behavior approximation is represented as a network of timed automata, and the authors show that their abstraction preserves the truthfulness of two real-time temporal logics.

In "Runtime Verification of Real-Time Event Streams Under Non-synchronized Arrival," Scheffel et al. present an approach to monitor concurrent systems with a shared clock abstracted as real-time event streams. The authors present a specification language and an online monitoring algorithm for the evaluation of the specifications.

In "Model Generation of Component-based Systems," Salva and Blot present an approach to learn the behavior of component-based systems from their execution traces using machine learning techniques. The resulting tool has been successfully trialed with network protocol traces.

The special section was announced during the track on Software Verification and Testing (SVT) at the 33rd ACM/SIGAPP Symposium On Applied Computing. One of the papers accepted in this special section is an extended version of a paper that originally appeared at the

✉ Yliès Falcone
ylies.falcone@univ-grenoble-alpes.fr

Leonardo Mariani
mariani@disco.unimib.it

[1] Univ. Grenoble Alpes, Inria, CNRS, Laboratoire d'Informatique de Grenoble, 38000 Grenoble, France

[2] University of Milano – Bicocca, Viale Sarca 336, IT-20126 Milan, Italy

SVT track. Collectively, these papers tackle cutting-edge topics thereby appealing to the experts in the field.

Finally, the guest editors thank the authors and the reviewers who contributed to the special section.

References

Bartocci, E., & Falcone, Y. (Eds.). (2018). *Lectures on runtime verification - introductory and advanced topics, lecture notes in computer science* (Vol. 10457). Springer. https://doi.org/10.1007/978-3-319-75632-5.

Bartocci, E., Falcone, Y., Bonakdarpour, B., Colombo, C., Decker, N., Havelund, K., Joshi, Y., Klaedtke, F., Milewicz, R., Reger, G., Rosu, G., Signoles, J., Thoma, D., Zalinescu, E., & Zhang, Y. (2019). First international competition on runtime verification: rules, benchmarks, tools, and final results of CRV 2014. *STTT, 21*(1), 31–70. https://doi.org/10.1007/s10009-017-0454-5.

Falcone, Y., Havelund, K., & Reger, G. (2013). A tutorial on runtime verification. In M. Broy, D. A. Peled, & G. Kalus (Eds.), *Engineering Dependable Software Systems, NATO Science for Peace and Security Series, D: Information and Communication Security* (Vol. 34, pp. 141–175). IOS Press. https://doi.org/10.3233/978-1-61499-207-3-141.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.