Formal Methods in Industrial Practice - Bridging the Gap (Track Summary)

Michael Felderer[1,2], Dilian Gurov[3], Marieke Huisman[4(✉)], Björn Lisper[5], and Rupert Schlick[6]

[1] University of Innsbruck, Innsbruck, Austria
[2] Blekinge Institute of Technology, Karlskrona, Sweden
[3] KTH Royal Institute of Technology, Stockholm, Sweden
[4] University of Twente, Enschede, The Netherlands
m.huisman@utwente.nl
[5] Mälardalen University, Västerås, Sweden
[6] AIT Austrian Institute of Technology, Seibersdorf, Austria

1 Motivation and Goals

Already for many decades, formal methods are considered to be the way forward to help the software industry to make more reliable and trustworthy software. However, despite this strong belief, and many individual success stories, no real change in industrial software development seems to happen. In fact, the software industry is moving fast forward itself, and the gap between what formal methods can achieve, and the daily software development practice does not seem to get smaller (and might even be growing).

In the past, numerous recommendations have already been made and studies performed on how to develop formal methods research in order to close the gap (e.g., [3,5–7,9,13,17]) between research and industrial practice, which also exists in other areas of software engineering like testing [10]. This track had the goal to investigate why the gap between research and industrial practice nevertheless still exists for formal methods, and what steps can be taken by the formal methods research community to bridge it.

The track consisted of three sessions of three speakers of 30 min each, followed by a 90 min closing discussion. We invited speakers that have collaborated with industry, and asked them for their experiences and recommendations on what should be done to close the gap. We also invited industrial speakers who have collaborated with academia, so as to learn from their experiences. Finally, the 4th session presented the idea to have a repository with formally verified benchmarks, to foster the industrial adoption of formal methods. During the closing discussion, we discussed the set up of such a benchmark. In addition, we also investigated if there are shared recommendations, and how we can put these recommendations into practice.

The track was in part a continuation of a Lorentz workshop in 2015, titled *Verification of Concurrent and Distributed Software: Towards Industrial Use.*

T. Margaria and B. Steffen (Eds.): ISoLA 2018, LNCS 11247, pp. 77–81, 2018.
https://doi.org/10.1007/978-3-030-03427-6_10

2 Contributions

2.1 Session 1: Testing and Requirements in Industrial Practice

The first session investigated current practices in testing and requirements engineering in industrial practice, and how formal techniques can help. During this session, the following papers were presented.

Peleska et al. [15] (*Model-based Testing for Avionic Systems Proven Benefits and Further Challenges*) report on the transition of model-based testing (MBT) from a widely discussed research discipline to an accepted technology that is currently becoming state of the art in industry, and in particular, in the field of safety-critical systems testing. They review how focal points of MBT-related research in the past have "survived" and found their way into today's commercial MBT products. The authors describe the benefits of MBT that are – from their experience – most appreciated by practitioners. Moreover, some interesting open challenges are described, and potential future solutions are presented. Their material is based on practical experience with recent MBT campaigns performed for Airbus in Germany.

Bardin et al. [2] (*Test Case Generation with PathCrawler/LTest: How to Automate an Industrial Testing Process*) consider automatic white-box testing based on formal methods as a relatively mature technology for which operational tools are available. Despite this, and the cost of manual testing, the technology is still rarely applied in an industrial setting. The authors describe how the specific needs of the user can be taken into account in order to build the necessary interface with a generic test tool. They present PathCrawler/LTest, a generator of test inputs for structural coverage of C functions, and describe the essential participation of the research branch of an industrial user in bridging the gap between the tool developers and their business unit and adapting PathCrawler/LTest to the needs of the latter.

Alzuhaibi et al. [1] (*Pitfalls upon Applying Model Learning to Industrial Legacy Software*) address refactoring of legacy software as one of the most common struggles of the current software industry, being costly and yet essential. They tackle this problem by applying model learning with the aim of understanding the observable behaviour of legacy components. The used technique interacts with a component in runtime and extracts abstract models that lead to better informed development decisions. The authors describe their experience in applying model learning to legacy software, aiming to prepare the newcomer for what shady pitfalls lie therein, as well as to provide the seasoned researcher with concrete cases and open problems. They narrate their experience in analysing certain legacy components at Philips Healthcare describing challenges faced, solutions implemented, and lessons learned.

2.2 Session 2: Software Verification in Industrial Practice

The second session then took the point of view of people working in software verification, and how they considered that their techniques could be used in industrial practice. During this session, the following papers were discussed.

Nyberg et al. [14] (*Formal Verification in Automotive Industry: Enablers and Obstacles*) describe and summarize their experiences from six case studies in applying formal verification techniques to embedded, safety-critical code. The studies have been conducted at Scania over the period of eight years. Despite certain successes, the authors admit to have so far failed to introduce formal techniques on a larger scale. Based on their experiences, they identify and discuss some key obstacles to, and enabling factors for, the successful incorporation of formal verification techniques into the software development and quality assurance process.

Knüppel et al. [11] (*Scalability of Deductive Verification Depends on Method Call Treatment*) address the problem of treating method calls in the context of deductive verification of safety-critical and security-critical applications applied in industry. During verification, a method call can either be replaced by an available method contract (called contracting) or by inlining the method's implementation. The authors argue that neither approach alone is feasible for verifying real-world software systems: Only relying on method inlining does not scale, as the number of inlined methods may lead to a combinatorial explosion; on the other hand, contracting is notoriously hard and time-consuming, making it economically unrealistic to be used exclusively. The authors discuss circumstances in which one of the two approaches is preferred. They evaluate the program verifier KeY with large programs varying in the number of method calls of each method and the maximum depth of the stack trace. Their analyses show that specifying 10% additional methods in a program can reduce the verification costs by up-to 50%, and, thus, an effective combination of contracting and method inlining is indispensable for the scalability of deductive verification.

Cok [8] (*Java Automated Deductive Verification in Practice: Lessons from industrial proof-based projects*) considers automated proof-based deductive verification used in industry to give confidence in the security and correctness of libraries and applications. The author presents various observations on current tools and processes based on recent experience with verification projects on industrial software, related to scalability, breadth, specification language expressibility and semantics, capabilities of underlying SMT tools, and integration into industrial build and continuous integration processes.

2.3 Session 3: Application Areas

The 3rd session investigated how focusing on specific application areas could help to make the use of formal verification techniques more feasible. During this session, the following papers were presented.

Bolignano and Plateau [4] (*Security Filters for IoT Domain Isolation*) consider network segregation as the key to the security of the Internet of Things, but also to the security of more traditional critical infrastructures or SCADA systems that need to be more and more connected and allow for remote operations. The authors believe that traditional firewalls or data diodes are not sufficient, considering the new issues at stake and that a new generation of filters is needed to replace or complement existing protections in these fields.

Larsen et al. [12] (*20 Years of Uppaal Enabled Industrial Model-Based Validation and Beyond*) review how the Uppaal Tool Suite served in industrial projects and was both driven and improved by them throughout the last 20 years. They show how the need of industry for model-based validation, performance evaluation and synthesis shaped the tool suite and how the tool suite aided the use cases it was applied in. The authors highlight a number of selected cases, including success stories and pitfalls, and discuss the important roles of both basic research and industrial projects.

Zakharov and Novikov [18] (*Verification of Operating System Monolithic Kernels without Extensions*) observe that operating systems and, in turn, applications strongly depend on monolithic kernels, and so the requirements for functionality, security, reliability and performance of the latter are ones of the highest. Currently used approaches to software quality assurance help to reveal quite many defects in monolithic kernels, but none of them aims at detecting all violations of checked requirements and providing some guaranties that target programs always operate correctly. The authors present a new method which is based on software verification and which enables thorough checking and finding complicated faults for various versions of monolithic kernels. One of its most important features is that it is not necessary to spend considerable effort for configuring tools and developing specifications to obtain valuable verification results, but one is able to steadily improve their quality. The authors implemented the suggested method within the software verification framework Klever and evaluated it on subsystems of the Linux monolithic kernel.

2.4 Session 4: A Repository of Formal Methods Examples and Experiments

Schlick et al. [16] (*A Proposal of an Example and Experiments Repository to Foster Industrial Adoption of Formal Methods*) observe that formal methods have been around almost since the beginning of computer science. Nonetheless, the perception in the formal methods community is that pickup by industry is rather low, measured by the potential benefits. As one approach to address this issue, they sketch the setup of a repository of software development problems and an accompanying open data storage to document, disseminate and compare solutions from formal model based methods. The purpose of this is to allow the industry to better understand the available solutions and more easily select and adopt one fitting their needs. At the same time, it should foster the adoption of open data and good scientific practice in the research field.

References

1. Alzuhaibi, O., Mooij, A., van Wezep, H., Groote, J.F.: Pitfalls upon applying model learning to industrial legacy software. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 121–138. Springer, Heidelberg (2018)

2. Bardin, S., Kosmatov, N., Marre, B., Mentré, D., Williams, N.: Test case generation with PathCrawler/LTest: how to automate an industrial testing process. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 104–120. Springer, Heidelberg (2018)
3. Bicarregui, J., et al.: Formal methods into practice: case studies in the application of the B method. IEE Proc.-Softw. **144**(2), 119–133 (1997)
4. Bolignano, D., Plateau, F.: Security filters for IoT domain isolation. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 194–211. Springer, Heidelberg (2018)
5. Bowen, J.P., Hinchey, M.G.: Ten commandments of formal methods. Computer **28**(4), 56–63 (1995)
6. Bowen, J.P., Hinchey, M.G.: Ten commandments revisited: a ten-year perspective on the industrial application of formal methods. In: Proceedings of the 10th International Workshop on Formal Methods For Industrial Critical Systems, pp. 8–16. ACM (2005)
7. Clarke, E.M., Wing, J.M.: Formal methods: state of the art and future directions. ACM Comput. Surv. (CSUR) **28**(4), 626–643 (1996)
8. Cok, D.: Java automated deductive verification in practice: lessons from industrial proof-based projects. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 176–193. Springer, Heidelberg (2018)
9. Davis, J.A., et al.: Study on the barriers to the industrial adoption of formal methods. In: Pecheur, C., Dierkes, M. (eds.) FMICS 2013. LNCS, vol. 8187, pp. 63–77. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41010-9_5
10. Garousi, V., Felderer, M.: Worlds apart: industrial and academic focus areas in software testing. IEEE Software **5**, 38–45 (2017)
11. Knüppel, A., Thüm, T., Padylla, C., Schaefer, I.: Scalability of deductive verification depends on method call treatment. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 159–175. Springer, Heidelberg (2018)
12. Larsen, K.G., Lorber, F., Nielsen, B.: 20 years of Uppaal enabled industrial model-based validation and beyond. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 212–229. Springer, Heidelberg (2018)
13. Margaria, T., Steffen, B.: Agile IT: thinking in user-centric models. In: Margaria, T., Steffen, B. (eds.) ISoLA 2008. CCIS, vol. 17, pp. 490–502. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88479-8_35
14. Nyberg, M., Gurov, D., Lidström, C., Rasmusson, A., Westman, J.: Formal verification in automotive industry: Enablers and obstacles. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 139–158. Springer, Heidelberg (2018)
15. Peleska, J., Brauer, J., ling Huang, W.: Model-based testing for avionic systems proven benefits and further challenges. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 82–103. Springer, Heidelberg (2018)
16. Schlick, R., Felderer, M., Majzik, I., Nardone, R., Raschke, A., Snook, C., Vittorini, V.: A proposal of an example and experiments repository to foster industrial adoption of formal methods. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 249–272. Springer, Heidelberg (2018)
17. Woodcock, J., Larsen, P.G., Bicarregui, J., Fitzgerald, J.: Formal methods: practice and experience. ACM Comput. Surv. (CSUR) **41**(4), 19 (2009)
18. Zakharov, I., Novikov, E.: Verification of operating system monolithic kernels without extensions. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 230–248. Springer, Heidelberg (2018)