Guest editorial: special section on software analysis, evolution, and reengineering

Selected papers from the SANER 2018 conference

Massimiliano Di Penta¹ • David C. Shepherd²

Published online: 19 February 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

This special section features selected, extended papers from the 25th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2018), held in Campobasso, Italy, March 20–23, 2018.

SANER is the premier event on the theory and practice of recovering information from existing software and systems. The event explores innovative methods to extract the many kinds of information that can be recovered from software, software engineering documents, and systems artifacts, and examines innovative ways of using this information in system renewal and program understanding. The SANER conference series started in 2015, from the merger of the Working Conference on Reverse Engineering (WCRE) and the European Conference on Software Maintenance and Reengineering (CSMR).

The SANER 2018 research track received 151 submissions, of which 38 were accepted and presented at the conference. This special section contains four extended versions of distinguished papers selected from those presented at the conference. All four papers underwent a completely new peer-reviewing process, which ensured the paper's technical soundness, relevance to the EMSE topics, and a sufficient delta with respect to the previously-published SANER 2018 paper. The four papers cover different topics of software evolution, including the quality of test reports, the use of static analysis tools, API discussions and breaking changes.

In the paper “A Systemic Framework for Crowdsourced Test Report Quality Assessment” Chen et al. deal with the problem of test report quality, which undermines inspection efficiency. They propose TEst Report Quality Assessment Framework (TERQAF), a framework that helps developers to determine whether, in a context where resources are limited, a test report should be inspected or not. This is done by defining a series of quantifiable

✉ Massimiliano Di Penta
dipenta@unisanno.it

David C. Shepherd
shepherd@vcu.edu

¹ University of Sannio, Via Traiano, 9, Benevento, Italy

² Virginia Commonwealth University, Richmond, VA, USA

indicators for quality test reports. In their empirical evaluation, Chen et al. show how TERQAF can help developers to handle test reports more efficiently.

In the paper “How Developers Engage with Static Analysis Tools in Different Contexts” Vassallo et al. investigate how the usage of Automatic Static Analysis Tools (ASATs) varies in different contexts. The investigation is performed in multiple stages, first through interviews and surveys, and then through manual inspection of configuration and build files from open source projects. The study shows how the relevance of ASATS varies depending on the project and domain, and highlights the need to improve strategies for the selection and prioritization of ASAT warnings.

In the paper “CAPS: A Supervised Technique for Classifying Stack Overflow Posts Concerning API Issues” Ahasanuzzaman et al. propose an approach to classify API-related sentences from Stack Overflow. First, they use a supervised learning approach using Conditional Random Field to identify API issue-related sentences. Then, they create a logistic regression model, accounting for the output of the CRF as well as for other features related to the post or to its author. The empirical evaluation shows how the proposed approach (CAPS) outperforms state-of-the-art techniques.

In the paper “You Broke My Code: Understanding the Motivations for Breaking Changes in APIs” Brito et al. study the phenomenon of API breaking changes. First, the authors performed a field study on the evolution of popular Java libraries, by also asking developers for the reasons of API breaking changes. Then, they complemented such an analysis by mining Stack Overflow discussions. The study results suggest that API breaking changes are mostly due to the need for implementing new features and improve maintainability. Finally, the authors provide suggestions to various stakeholders on how to deal with API breaking changes.

We would like to thank the authors for submitting high-quality papers, and the reviewers for providing timely, constructive and detailed feedback. Finally, we sincerely hope that the readers will enjoy the four papers and find them inspiring for their research and practice.

Massimiliano Di Penta and David C. Shepherd

Guest Editors of the special section on software analysis, evolution, and reengineering.

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

Massimiliano Di Penta is a full professor at the University of Sannio, Italy. His research interests include software maintenance and evolution, mining software repositories, empirical software engineering, search-based software engineering, and software testing. He is an author of over 260 papers that appeared in international journals, conferences, and workshops. He has received several awards for research and service, including four ACM SIGSOFT Distinguished paper awards. According to a recent article that appeared in the JSS journal, he was considered the second-most productive expert researcher in the software engineering area for the period 2010–2017. He serves and has served in the organizing and program committees of more than 100 conferences, including ICSE, FSE, ASE, and ICSME. He will be program co-chair of ESEC/FSE 2021 and of ICSE 2023. He is co-editor in chief of the *Journal of Software: Evolution and Processes* edited by Wiley, editorial board member of *ACM Transactions on Software Engineering and Methodology* and *Empirical Software Engineering Journal* edited by Springer, and has served the editorial board of the *IEEE Transactions on Software Engineering*.

David C. Shepherd is an Associate Professor in the Department of Computer Science at Virginia Commonwealth University. He earned his Ph.D. and M.S. in Computer Science at the University of Delaware, and his B.S. in Computer Science at Virginia Commonwealth University. David has since worked as a postdoctoral fellow in the Department of Computer Science at the University of British Columbia, built sweat equity as employee #9 at Tasktop Technologies, and risen to Senior Principal Scientist at ABB Corporate Research. His research has produced tools that have been used by thousands, innovations that have been featured in the popular press, and practical ideas that have won business plan competitions. Dr. Shepherd currently serves as the Co-Editor-in-Chief of the *Journal of Systems & Software*. He has served as the Program Co-Chair for the industrial tracks of the 20th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2017), the 39th ACM/IEEE International Conference on Software Engineering (ICSE 2017), and the 31st International Conference on Software Maintenance and Evolution (ICSME 2015). His current work focuses on enabling end-user programming for industrial machines and increasing diversity in computer science.