Preface to the special issue on program comprehension

Janet Siegmund¹ · Chanchal K. Roy²

Published online: 13 April 2020 Published online: 13 April 2020
© The Author(s) 2020, corrected publication 2021

We are excited to present six selected papers of the 26th IEEE/ACM International Conference on Program Comprehension 2018, which took place in Gothenburg, Sweden, together with the 40th International Conference on Software Engineering. We received 69 submissions in total, of which we could accept 26. Each paper received at least three reviews and was discussed online, following a triple blind model, such that the reviewers did not know the identities of the authors, and that reviewers even did not know the identities of the other reviewers.

The PC chairs selected papers to be invited for the special issue, such that all nominees for a distinguished paper award were invited. Furthermore, papers with a positive average score (1.0 on a 4 point scale from −2 to 2) and discussions among the PC members were also considered, as well as suggestions from the PC members. This resulted in the invitation of six papers, which all could be accepted for publication after considerable extension according to EMSE standard.

This first invited paper, which received a distinguished paper award, evaluated the cognitive load of developers. The author team of Sarah Fakhoury, Devjeet Roy, Yuzhan Ma, Venera Arnaoudova, and Olusola Adesope contribute an extended version entitled “Measuring the Impact of Lexical and Structural Inconsistencies on Developers’ Cognitive Load during Bug Localization”. The paper presents a multi-modal approach to assess developers cognitive load based on a combination of functional near-infrared spectroscopy (fNIRS) and eye tracking. In addition to demonstrating the reliability of their multimodal approach by comparing the self-estimated cognitive load of participants with sensor information, the authors found evidence that changing the structure of code (e.g., violating coding conventions) does not increase cognitive load, but violating naming conventions for identifiers does. Additionally, the modalities to assess cognitive load all seem to capture different aspects of task difficulty.

The second invited paper also received a distinguished paper award. The author team of Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin contribute their extended version entitled “Deep Code Comment Generation with Hybrid Lexical and Syntactical Information”. In their paper, the authors develop an approach to automatically generate comments to Java methods,

✉ Janet Siegmund

¹ Chemnitz University of Technology, Chemnitz, Germany

² Department of Computer Science, University of Saskatchewan, Saskatoon, Canada

thereby making it easy to keep the comments up-to-date with implementation or add comments to existing methods. To this end, the authors train a deep neural network that combines lexical and structural information of Java methods. Their extensive evaluation on almost 10,000 Github projects shows the wide applicability of the newly developed approach, which is named Hybrid-DeepCom.

The paper by Carmine Vassallo, Sebastian Proksch, Timothy Zemp, and Harald Gall regarding broken builds was also a candidate for a distinguished paper award. In their paper “Every Build You Break: Developer-Oriented Assistance for Build Failure Resolution”, the authors analyze build failures and summarize possible reasons in their tool Bart. Additionally, Bart suggests possible fixes for build failures based on solutions from the Internet. In a case study, developers could better understand reasons for build failures and fix them faster. However, certain build failures are difficult to understand and fix, especially those related to infrastructure problems or inadequate log information.

The next invited paper by Stefanie Beyer, Christian Macho, Massimiliano Di Penta, and Martin Pinzger entitled “What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories” focused on how Stack Overflow can support developers in finding information. Typically, questions on Stack Overflow are categorized based on technological aspects, such as the programming language of a post. However, the purpose of a question is also helpful to understand the context, for example, to discuss bugs, proper API usage, or solicit opinions about a technology. Thus, the authors developed an approach to automatically classify posts depending on the purpose of a question. Based on a ground truth of 1000 manually classified posts, the authors build a classifier that can automatically detect the purpose of a post, thereby supporting developers in finding a suitable post to a question.

The author team Shanshan Li, Xu Niu, M.D., Zhouyang Jia, Xiangke Liao, Ji Wang, and Tao Li contributed their paper “Guiding log revisions by learning from software evolution history”. They focused on log statements of code, which include runtime information of software to support bug-fixing activities. However, since code evolves, the logs also have to be updated to provide useful guidance. To support the update of logs, the authors developed an approach that analyzes previous log modifications to learn rules of the modification, and then the authors use these rules to suggest modifications to logs. In an empirical study of several open software projects, the authors demonstrated the usefulness of their approach, such that most of these suggested modifications could be accepted.

The paper “Third-Party Libraries in Mobile Apps: When, How, and Why Developers Update Them” by Pasquale Salza, Fabio Palomba, Dario Di Nucci, Andrea De Lucia, and Filomena Ferrucci analyzed how developers update third-party libraries. To this end, they used a two-step process, such that first, they analyzed over 2700 apps to understand the usage of third-party libraries, and they found that only 1% of the entire commit activity is related to an update of a third-party library. Second, they surveyed developers and found that the reasons for this reluctance lie in the high cost of updating a library (compared to the benefits) and the fear of breaking existing code. When developers do update a library, it is mostly to avoid bug propagation or ensure compatibility with a new Android version.

We found these papers to provide an inspired read and we hope that you will also enjoy them. May they inspire you to some great work to submit at the upcoming editions of the International Conference on Program Comprehension and become part of an open and thriving community.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Janet Siegmund is professor for Software Engineering at the Chemnitz University of Technology. Prior to that, she led the junior research group PICCARD, funded by the Centre Digitisation.Bavaria. She received her Ph.D. from the University of Magdeburg in 2012 and she holds two master's degrees, one in Computer Science and one in Psychology. In her research, she focuses on the human factor in software engineering, for example, when writing source code. Janet Siegmund is the co-author of more than 30 peer-reviewed journal, conference, and workshop publications. She regularly serves as program-committee member for conferences and workshops. From 2014 to 2017, she was in the steering committee of the International Conference on Program Comprehension.