Empir Software Eng (2017) 22:1438–1439
DOI 10.1007/s10664-017-9525-1

EDITORIAL

Guest editorial: Program comprehension

Rocco Oliveto[1] · **Christian Bird**[2]

Published online: 25 May 2017

Program comprehension is an important part of software engineering and is central to software maintenance, reuse, inspection, reverse engineering, reengineering, migration, and extension of existing software systems. Program comprehension is surprisingly complex and the past decade has seen a surge in research aimed at investigating, understanding, and aiding how developers reason about, ask and answer questions about, and make changes to software.

This special issue of the Empirical Software Engineering Journal contains extensions of the highest quality papers published at the International Conference on Program Comprehension in 2015 in Florence Italy and represents some of the leading research being conducted in this important area. Each paper was significantly extended and went through multiple rounds of peer review and revision prior to publication. We hope that readers find these papers useful and enjoyable.

The first paper "*How Programmers Read Regular Code: a Controlled Experiment Using Eye Tracking*" by Ahmad Jbara and Dror G. Feitelson uses eye tracking to explore the hypothesis that a developer can understand code faster if it contains patterns that he or she has seen and understood before. They were able to successfully model the time required to understand code similar to previously seen code using an exponential decay model. Their findings suggest that seeing repeated code does not lead to understanding more quickly, but rather that the initial code receives more focus and is looked at more times while the later instances can be skimmed.

The second paper "*Documenting and Sharing Software Knowledge Using Screencasts*" by Laura MacLeod, Margaret-Anne Storey, and Andreas Bergen examines the growing use of screencasts by developers as an alternate form of documentation. Through manual analysis of screencasts on YouTube and interviews of authors of screencasts to understand what kinds of knowledge is shared, the techniques used to share it, and the motivations for creating the screencasts. The authors explain how well screencasts work, how they are useful for developing an online reputation, and provide best practices that other developers can use to create high quality and useful screencasts.

✉ Rocco Oliveto
rocco.oliveto@unimol.it

Christian Bird
cbird@microsoft.com

[1] University of Molise, Pesche, Italy

[2] Microsoft Research, Redmond, USA

The third paper "*The Last Line Effect Explained*" by Moritz Beller, Andy Zaidman, Andrey Karpov, and Rolf A. Zwaan explores the hypothesis that when a small piece of code is copied from one location to another (a so-called micro-clone), the last line is much more likely to contain an error that other lines in the copied code. Through an investigation of code and interviews, they examine the underlying psychological mechanisms for the presence of these mistakes and find that they are often trivial errors. Their findings are beneficial in that they point developers to areas where mistakes are more likely. The authors also present a tool to find mistake-prone micro-clones.

The forth paper entitled "*License Usage and Changes: a Large-Scale Study on GitHub*" by Christopher Vendome, Mario Linares-Vasquez, Gabriele Bavota, Massimiliano Di Penta, Daniel German, and Denys Poshyvanyk reports on a large-scale empirical study investigating when and why developers adopt or change software licenses. Using both qualitative and quantitative approaches they examine the history of over 51,000 projects. Reasons for license adoption and changes include making it easier for code to be used in commercial contexts, requests from users to clarify what license is in use, and newer versions of licenses (such as the GPL) becoming available, and ensuring license compatibility. However, the authors also discovered that there is almost no traceability for when and why the license changes are made, which may be cause for concern and highlights the need for techniques to help in choosing or changing licenses as well as recording the rationale for the changes.

We hope you enjoy the papers in this special issue. We also hope that such papers will help you to acquire knowledge on this field and shed some lights on new research directions.

We would like to thank all the authors who extended their research and addressed the feedback from the external reviewers. We also appreciate the time and effort put in by the initial ICPC 2015 program committee as well as the reviewers of the extended versions of these papers that led to a successful special issue with high quality papers. Finally, we would like to thank the editorial board of the Journal of Empirical Software Engineering and the Editors in Chief Thomas Zimmermann, Lionel Briand, and Robert Feldt for help and guidance during creation of this special issue.