Editor: **Silvia Abrahão**
Universitat Politècnica de València
sabrahao@disc.upv.es

Editor: **Miroslaw Staron**
Chalmers University of Technology and University of Gothenburg
miroslaw.staron@cse.gu.se

Open Source Software: Communities and Quality

Silvia Abrahão, Miroslaw Staron, Alexander Serebrenik, Birgit Penzenstadler, and Rafael Capilla

FOLLOWING ALONG WITH the theme of this special issue of *IEEE Software*, this edition of the "Practitioners' Digest" reports on papers about open source software from the 44th ACM/IEEE International Conference on Software Engineering (ICSE 2022); the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2022); the 15th IEEE International Conference on Software Testing, Verification and Validation (ICST 2022); and the 19th International Conference on Mining Software Repositories (MSR 2022). Feedback or suggestions are welcome. In addition, if you try or adopt any of the practices included in the column, please send us and the authors of the paper(s) a note about your experiences.

Toxicity in Open Source Discussions

Have you ever been on the receiving end of entitled, demanding, and arrogant comments in a software project? They can come from project users having a bad day (or year), or they could be insults arising from technical disagreements. In the paper "Did you Miss My Comment or What? Understanding Toxicity in Open Source Discussions," Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian Kästner curate a sample of 100 toxic GitHub issue discussions combining multiple search and sampling strategies. While online toxicity is ubiquitous across the Internet, and its negative impact has been well documented, toxicity manifests differently on various platforms. Specifically, toxicity in open source communities, while frequently discussed, is not well understood. The authors make an attempt at understanding the characteristics of open source toxicity to better inform future work on designing effective intervention and detection methods.

Main triggers that caused toxicity were failure in using the tool, code or error message, technical disagreements, politics or ideology, or past interactions. A key finding of this study is that entitled comments are a frequent source of toxicity (25 issues in their sample). These types of comments make demands of people or projects as if the author had an expectation due to a contractual relationship or payment (e.g., insisting their requests are given priority or demanding bugs to be resolved quickly). This represents a form of toxicity not previously observed as a common problem in other platforms.

The conclusions drawn from this study were that 1) toxicity presents differently on GitHub, 2) open source experience does not prevent toxicity, 3) research is needed on the harm cause by toxicity, 4) this presents opportunities to build open-source-specific detectors, and 5) tailored detection-based interventions are promising. This paper appears in the proceedings of ICSE 2022. Access it at https://tinyurl.com/4hr839db.

Newcomers in Open Source Software Projects

It is well known that onboarding new developers in an open source project is a challenging task: despite the very best attempts of the project maintainers, the newcomers still experience numerous barriers during this process. One plausible explanation of this phenomenon is the misalignment of the expectations of the newcomers, more experienced contributors, and project maintainers, and this is exactly the focus of the study of Fabio Santos, Bianca Trinkenreich, João Felipe Pimentel, Igor Wiese, Igor Steinmacher, Anita Sarma, and Marco A. Gerosa. These authors started by interviewing 17 maintainers to understand how newcomers choose an

Digital Object Identifier 10.1109/MS.2023.3270779
Date of current version: 14 July 2023

0740-7459/23©2023IEEE

issue and how the community can help. The issue selection strategies identified in these interviews served as a basis for a follow-up survey of maintainers, frequent contributors, and newcomers. A total of 64 survey respondents were asked to rank the importance of the strategies. The findings show that the opinions about the relative importance of various strategies diverge between the three groups of respondents. In particular, newcomers highlight the importance of technical support (e.g., related to setting up the environment), while experienced contributors and maintainers stress the importance of the conceptual understanding of an issue. Furthermore, supporting the onboarding of newcomers was deemed important both by the newcomers and by the experienced developers but, surprisingly, not by the maintainers. In fact, maintainers ranked this strategy last. These authors consider this surprising since the maintainers are expected to prioritize the onboarding process to count on human resources to work on the issues. In summary, the findings suggest gaps in the perspectives of newcomers, experienced contributors, and maintainers; such gaps are likely to lead to divergent expectations of different groups of contributors, hindering the onboarding of the newcomers. Hence, open source software communities interested in better onboarding newcomers should prioritize the strategies considered the most important by newcomers themselves. This paper appears in the proceedings of ESEM 2022. Access it at https://tinyurl.com/yy8p8a4v.

Quality of Open Source Ansible Scripts

Infrastructure as code and Ansible have become an important technology for automating the deployment of software in professional IT environments. Despite their popularity, these types of programs, often called Ansible scripts, are usually treated differently from the software product code. Quality assurance is an example of this kind of different treatment. In the paper "As Code Testing: Characterizing Test Quality in Open Source Ansible Development," Mohammad Mahedi Hassan and Akond Rahman present an empirical study of Ansible scripts in open source projects. The goal of the study is to understand and categorize the most common problems in these special kinds of programs. The study finds that 5.6% of the analyzed commits are related to defects in Ansible scripts. The paper lists seven categories of defects in analyzed Ansible test scripts: configuration, dependency, idempotency, logging, performance, security, and style. The largest categories are configuration and performance. The paper also found that there are three testing patterns that correlate with the defects in Ansible scripts: assertion roulette, local only testing, and remote mystery. In addition to identifying and classifying patterns of defects and tests, the paper provides a tool, Test Pattern Miner for Ansible, which can be used by practitioners to analyze their own scripts, reduce the number of defects and increase the quality of their software. This paper appears in the proceedings of ICST 2022. Access it at https://tinyurl.com/mvdhmz6c.

> The most common challenges faced by developers when reviewing refactorings are quality, refactoring (correctness), objective, testing, integration, and management of the review activity.

Analyzing Code Review Practices

"Code Review Practices for Refactoring Changes: An Empirical Study on OpenStack" by Eman Abdullah AlOmar, Moataz Chouchen, Mohamed Wiem Mkaouer, and Ali Ouni starts by setting the scene that modern code reviews are widely used to improve software quality, both in open source projects and in industrial projects. The authors study how developers decide whether to accept or reject a refactoring change. The authors analyze over 700,000 code changes from OpenStack projects; this include 5,505 reviews related to refactorings. The results show that refactoring-related reviews involve more reviewers and contain more review comments as well as having

a lengthier review time (among others). The most common challenges faced by developers when reviewing refactorings are quality, refactoring (correctness), objective, testing, integration, and management of the review activity. In addition to analyzing the review comments, the paper identifies a few implications for practitioners and researchers; for example, it recommends establishing guidelines for refactoring-related reviews to increase the speed and quality of the code. This paper appears in the proceedings of MSR 2022. Access it at https://tinyurl.com/2u5u7w9r.

Orphan Vulnerabilities in Open Source Software

The key premise of open source software is that developers can and often do copy open source code into their projects. This practice may result in vulnerabilities that persist in the copied code even when discovered and fixed in the original project.

Such vulnerabilities are called orphan vulnerabilities in the paper "The Extent of Orphan Vulnerabilities From Code Reuse in Open Source Software" by David Reid, Mahmoud Jahanshahi, and Audris Mockus. The paper studies the reuse of vulnerable code across all open source software. First, the authors propose Vulnerability Detection in Open Source, a tool that, given a vulnerability fix in one project, identifies all other projects that contain either still vulnerable or fixed code. They leverage the world of code infrastructure to find file-level code duplication in any language from a nearly complete collection of open source software. Second, the authors conducted a case study to understand the issues related to the spread of four software security vulnerabilities from three libraries written in C and Go. The results of the study show that many active repositories have copied vulnerable code and that only rarely do such projects patch the vulnerabilities even after being provided a fix.

These results have implications for practice. In particular, the proposed tool can help maintainers and users of vulnerable projects understand the risks of the vulnerability in their code and warn users that want to reuse such code about the unpatched vulnerabilities. This paper appears in the proceedings of ICSE 2022. Access it at http://web.eecs.utk.edu/~dreid6/ICSE2022_Orphan.pdf.

ABOUT THE AUTHORS

SILVIA ABRAHÃO is an associate professor (accredited to full professor) at Universitat Politècnica de València, 46022 Valencia, Spain. Contact her at sabrahao@disc.upv.es.

MIROSLAW STARON is a professor in the Software Engineering Division, Chalmers University of Technology and the University of Gothenburg, SE-412 96 Gothenburg, Sweden. Contact him at miroslaw.staron@cse.gu.se.

ALEXANDER SEREBRENIK is a professor at Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands. Contact him at a.serebrenik@tue.nl.

BIRGIT PENZENSTADLER is an assistant professor at Chalmers University of Technology and the University of Gothenburg, 412 96 Gothenburg, Sweden. Contact her at birgitp@chalmers.se.

RAFAEL CAPILLA is a professor at Rey Juan Carlos University, 28933 Madrid, Spain. Contact him at https://www.linkedin.com/in/rafael-capilla-89b3857/ or rafael.capilla@urjc.es.

Community and Design Smells

"Analyzing the Relationship Between Community and Design Smells in Open Source Software Projects: An Empirical Study" by Haris Mumtaz, Paramvir Singh, and Kelly Blincoe investigates the relationship between community smells and design smells. Community smells refer to the suboptimal patterns in organizational and social structures of software teams, while design smells are deviations of good architecture patterns. Specifically, it examines the correlation and trend similarities between both types of smells across 10 large-scale Apache projects. The authors analyze the communication and interaction problems in software teams that may lead to community smells like "organizational silos" (i.e., isolated subgroups of developers do not communicate except through one or two of their members) and "missing links" (i.e., developers work in isolation without communication with their peers) related to the appearance of design smells. They use three statistical methods to empirically evaluate the possible correlation between community and design smells. The results reveal significant correlation and trend similarities of design smells with "missing links" compared with "organizational silos." Therefore, community smells have shown their relevance to prove the presence of certain design smells according to the behavior of open source development teams and how team restructuring improves the broken communication or suboptimal ways to communicate. The additional benefit obtained from this empirical study is to enhance communication patterns in open source developer communities. This paper appears in the proceedings of ESEM 2022. Access it at https://kblincoe.github.io/publications/2022_ESEM_Smells.pdf.

Digital Object Identifier 10.1109/MS.2023.3284234