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

Focusing on Developers in the Era of AI and ML

Silvia Abrahão, Miroslaw Staron, Alexander Serebrenik, Birgit Penzenstadler, Rafael Prikladnicki, and Henry Muccini

THIS EDITION OF the "Practitioners' Digest" covers recent articles and/or papers on novel approaches and tools to support developers and communities in the era of artificial intelligence from the Technical Track and the Software Engineering in Practice Track of the 45th IEEE/ACM International Conference on Software Engineering (ICSE 2023). 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.

Developers and Their Implementation of Design Decisions

In "A Qualitative Study on the Implementation Design Decisions of Developers," Jenny T. Liang, Maryam Arab, Minhyuk Ko, Amy Ko, and Thomas LaToza, study the specific decisions where developers select one specific way out of many alternatives to implement a behavior in code. That's what the authors call "implementation design decisions." Based on 46 survey responses and 14 semistructured interviews (half in North America, half in the rest of the world) with professional developers about their decision types, considerations, processes, and expertise, they find that implementation design decisions require constant monitoring of higher-level design choices, such as requirements and architecture, rather than being implied by them. Developers seem to have a consistent general structure to their implementation decision-making process, but no single process is exactly the same.

The findings imply insights on teaching developers how to make implementation design decisions as interpreting requirements throughout implementation is key to making these decisions. Furthermore, as implementation design decisions require rounds of iteration in different stages, mentors can encourage less experienced engineers to use prototyping as regular practice to learn from small-scale failures in iterations. This paper was presented at the Technical Track of the International Conference on Software Engineering (ICSE 2023). Access it at https://tinyurl.com/yck769du.

Augmenting Software with AI

Augmenting existing software products with artificial intelligence (AI) capabilities sounds easier than it really is. It requires the migrating organization to understand where the AI capabilities make the most sense, elicit requirements for these new capabilities and then introduce them. It also requires the migrating company to work alongside with their stakeholders and customers to ensure that these new capabilities bring value. In the paper "Challenges in Adopting Artificial Intelligence Based User Input Verification Framework in Reporting Software Systems," Dong Jae Kim, Tse-Hsun Chen, Steve Sporea, Andrei Toma, Laura Weinkam, and Sarah Sajedi, describe their experiences from adding machine learning for anomaly detection to a chemical production monitoring system. The machine learning components use unsupervised learning ensembles of models to identify anomalies in the reported data, with the goal to make the monitoring process more cost-effective and efficient.

In total, the paper identifies nine challenges, grouped into three categories—challenges in collaborative

Digital Object Identifier 10.1109/MS.2023.3302884
Date of current version: 1 December 2023

requirement engineering, involving many development teams of various backgrounds, challenges in AI adoption in user-centric reporting systems with stakeholders with diverse operational behavior and challenges in AI verification, and challenges in user-centric reporting systems with stakeholders with diverse patterns of operational behavior and company-dependent anomalies. In addition to such challenges as requirements for training data, the paper identifies practical challenges such as when and how to relabel and annotate new data when the process changes or how to deal with requirements from different end-users who want different time scales for anomaly detection (yearly versus weekly). The authors also discuss challenges related to scaling the process while keeping a human-in-the-loop approach for potential validation. This work's practical implication is the discussion on the challenges as well as their adoption and avoidance strategies; they will help organizations who are considering adopting AI technologies in their existing software products. The paper was presented at the Software Engineering in Practice Track of the International Conference on Software Engineering (ICSE 2023). Access it at https://tinyurl.com/msvzdc8y.

AI Supporting Modern Code Reviews

Code reviews are certainly the activity where modern machine learning can make a big difference, especially with the introduction of large language models like GPT-4. In their paper "Using Large-scale Heterogeneous Graph Representation Learning for Code Review Recommendations at Microsoft," Jiyang Zhang, Ram Bairi, Christian Bird, Ujjwal Raizada, Apoorva Agrawal, Yamini Jhawar, Kim Herzig, and Arie van Deursen, study how to improve the process of automatically selecting relevant code reviewers. The current approaches are based on simplistic heuristics of historical code reviews (e.g., the best reviewer is the one who reviewed the same file in the past), but this new approach is based on a graph of social interactions between developers. The tool presented in this paper uses graph neural networks to 1) successfully model the existing process of selecting reviewers (with 73% accuracy) and 2) successfully find new reviewers who the existing process missed. The authors did this by analyzing a half year history of 332 repositories from Microsoft. The analyses included both the history and also the semantics (and context) of pull requests, using a heuristic that if a reviewer contributed positively to a pull request that was semantically similar to the current one, the reviewer can be a good candidate for the current pull request. This study shows that using advanced machine learning can provide tremendous help to software developers without replacing them, which is a great way to increase the life of software engineers. The paper was presented at the Software Engineering in Practice track of the International Conference on Software Engineering (ICSE 2023). Access it at https://tinyurl.com/mwhyv4pj.

Code Search Tools

The paper "You Don't Know Search: Helping Users Find Code by Automatically Evaluating Alternative Queries" by Rijnard van Tonder tackles the difficulties associated with creating a code search query language that caters to a diverse range of users. To achieve this, the authors introduce a novel technique named automated query evaluation (AQE). AQE generates and dynamically executes different query interpretations in situations where users may experience frustration. Through an A/B test, the paper assesses AQE and discovers that its activation leads to an increased likelihood of users clicking on search results. The aim is to enhance code search tools, and the authors recommend further research and development of AQE for the future.

The main result shows that relative to the control group, users are on average 22% more likely to click on a search result at all on any given day when AQE is active. The paper observes that AQE triggers slightly more often when the original query produces some results (52%) versus no results (48%). This suggests that AQE provides greater utility when the original query produces no results, since it creates an opportunity for users to see or click on results that they would not have had otherwise.

Overall, the paper provides practical insights and solutions for improving code search tools and making it easier for users to find relevant code. The paper was presented at the Software Engineering in Practice Track of the International Conference on Software Engineering (ICSE 2023). Access it at https://arxiv.org/abs/2212.03459.

Psychoactive Substances and Developers

The paper "From Organizations to Individuals: Psychoactive Substance Use By Professional Programmers" by Kaia Newman, Madeline Endres, Westley Weimer, and Brittany

Johnson examines the use of psychoactive substances by professional programmers. Such substances influence the brain to alter perceptions and moods and have the potential to have positive and negative effects on critical software engineering tasks. They are widely used in software, but that use is not well understood. The authors conducted a qualitative investigation based on 26 interviews and thematic analysis in a first qualitative study like this. The authors do a great job by explaining the individual substance use motivations and impacts, such as mental health considerations as well as substance use and productivity (including per-substance and per-task breakdowns). They also discuss the socialization effects of substance use in software, such as the impact on soft skills, visible work use and stigma, and the effect of remote work and the impact on organizational policy. Some key findings include that attention deficit hyperactivity disorder (ADHD) is the primary mental health diagnosis driving substance use in the sample. The majority of those diagnosed cited their seeking diagnosis was at least in part due to symptoms present in their software work. Stimulants prescribed as a result of diagnosis are viewed to have a positive impact on their software development.

The paper also found that when using substances for programming enhancement, increasing productivity is the most common goal, especially when it comes to using stimulants. Increasing creativity, work quality, or work enjoyment are cited less commonly, though when they are, it is usually in the context of alcohol, cannabis, or psychedelics. Developers choose to use different substances for different software tasks (e.g., stimulants for debugging, cannabis for brainstorming), evidence that developers self-regulate their substance use, informally using it analogously to other development tools. Finally, the study found that alcohol and prescription stimulants are more likely to be used and discussed than other psychoactive substances. Cannabis and psychedelics are more taboo. Both workplace productivity culture and work-from-home policies can be associated with increases in substance use.

Some key conclusions of this work can be considered both at the individual level—alleviating mental health symptoms or desired programming enhancement are the primary motivations for using psychoactive substances, socialization level—participants describe a positive impact on "soft" skills, as well as visible use at work for many substances (and increased use under work from home), and the organization level—there is widespread agreement that antidrug policies are unclear and ineffective. Such policies are viewed as indicative of corporate culture and may have a negative impact on hiring and retention.

An important observation is the use of substances that are illegal or may be dangerous in some contexts, the authors make it clear that they neither endorse nor condemn this behavior.

An important observation is the use of substances that are illegal or may be dangerous in some contexts, the authors make it clear that they neither endorse nor condemn this behavior. Rather, the goal is to understand, present, and qualitatively analyze the lived experiences of psychoactive substance users working with software. The paper was presented at the Technical Track of the International Conference on Software Engineering (ICSE 2023). Access it at https://tinyurl.com/uu688rnt.

Virtual Communities and Belonging

As our society relies on open source software (OSS), significant efforts are being made by both researchers and practitioners to ensure the sustainability of open source communities. The sense of virtual community, which refers to the feeling of belonging that members have toward others within a certain group, such as a group of developers working on the same open source project, can influence the sustainability of a community, as well as the satisfaction and involvement of contributors. This is why Bianca Trinkenrech, Klaas-Jan

Stol, Anita Sarma, Daniel M. German, Marco A. Gerosa, and Igor Steinmacher have investigated how a sense of virtual community develops in OSS projects in the paper "Do I Belong? Modeling Sense of Virtual Community Among Linux Kernel Contributors."

Based on an analysis of the Linux Kernel developer community ($N = 225$), the authors observe that intrinsic motivations, such as the feeling of kinship or the desire to increase fun and pleasure, are positively associated with a sense of virtual community. However, living in an authoritative country and being paid to contribute can reduce the sense of virtual community. The study results suggest that in order to retain contributors and increase their sense of virtual community, open source projects should go beyond offering online interest groups, chat rooms, and discussion forums. Instead, they should employ online tools that provide shared spaces for contributors to work "together" and improve interaction between them. OSS communities should also encourage the exchange of support among members. Similarly, newcomers should be encouraged to engage in communication and initiate social connections. Implementing these suggestions is expected to foster a stronger sense of virtual community and, consequently, increase the sustainability of the Linux Kernel community. The paper was presented at the Technical Track of the International Conference of Software Engineering (ICSE 2023). Access it at https://tinyurl.com/49fkd378.

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 PRIKLADNICKI is an associate professor in the School of Technology, Pontifical Catholic University of Rio Grande do Sul, Porto Alegre 90450-171, Brazil. Contact him at https://www.inf.pucrs.br/rafael/ or rafaelp@pucrs.br.

HENRY MUCCINI is a full professor of software engineering at the University of L'Aquila, 67100 L'Aquila, Italy. Contact him at henry.muccini@univaq.it.