Automated Authoring, Onboarding Developers, and Extracting Decision Rationale

Jeffrey C. Carver, Rafael Capilla, Alexander Serebrenik, and Miroslaw Staron

THE “PRACTITIONERS’ DIGEST” department in this issue of IEEE Software includes papers from the 2021 International Conference on Software Engineering (ICSE). Feedback or suggestions are welcome. In addition, if you try or adopt any of the practices included in this article, please send me and the authors of the paper(s) a note about your experiences.

Automated Authoring of Methods
“Siri, Write the Next Method,” by Fengcai Wen and colleagues, expands upon the long-standing concept of code completion to include the automation of function signatures and function bodies. Code completion, an important support mechanism used by professional developers, helps complete method names, suggests the signature of an application programming interface function, or simply formats code. FeaRS, the authors’ new tool, takes that concept one step further by writing entire function signatures and function bodies based on the analysis of 2.5 million commits of existing functions in various open source projects. The evaluation of FeaRS showed its ability in providing highly relevant recommendations for developers of Android apps but also highlighted the limitations of the tool in supporting a wide diversity of implementation scenarios. The best results are obtained for small techniques, with some promising results for GUI-related methods. The authors also have created a plug-in for Android Studio, which should help improve the productivity of professional software engineers. This paper appears in the main research track at ICSE 2021. Access it at http://bit.ly/PD-2021-Sept-01.

Onboarding Developers
“A Case Study of Onboarding in Software Teams: Tasks and Strategies,” by An Ju and colleagues, reports on the results of a study with 32 developers and 15 managers involved in the onboarding process. Onboarding new team members requires patience and energy. The new team members need to understand the product, the organization, and the culture of the company. The current managers and team members must understand what the new team members need to be able to function well and to provide value to the product or service. This study identified three strategies that work well in this context: 1) Simple-Complex, when the new developers receive tasks that gradually increase in complexity; 2) Priority-First, when the tasks are assigned by matching priority with seniority; and 3) Exploration-Based, when the uncertain or undefined tasks are assigned to
senior developers to enable exploration of the product structure. The paper breaks down these strategies into tangible guidelines that can be used by other teams. For example, in the Simple-Complex strategy, the paper provides guidelines about three levels of complexity—bugs or configuration setting during the first week, simple features for the next two to three weeks, and more advanced bugs and features during the next three to nine months—eventually leading to expertise in the product. In addition to the case study results, the paper reports on a follow-up survey of 189 developers and 37 managers that found the results to be representative and the recommendations actionable. This paper appears in the main research track at ICSE 2021. Access the paper at http://bit.ly/PD-2021-Sept-02.

“Onboarding Versus Diversity, Productivity and Quality—Empirical Study of the OpenStack Ecosystem,” by Armstrong Foundjem and colleagues, describes a study of onboarding in OpenStack, a large-scale, open source software initiative comprising many independent but interrelated projects. To be successful, onboarding in such initiatives, known as ecosystems, should support the contributors in obtaining knowledge and mastering skills required by both the ecosystem as a whole as well as individual projects in which the contributor is interested. Based on an observational study of 72 contributors and 13 mentors, the authors have concluded that the mentors balance theoretical (for example, knowledge of the contribution workflow or the importance of active communication skills; 40%) and hands-on (60%) activities. They further identified primary challenges associated with onboarding: while contributors should be self-motivated and committed, the mentors should have a vast knowledge of the ecosystem and be ready to invest substantial effort in preparing the teaching material and keeping it up to date. The expected benefits of the onboarding program for the ecosystem are more diverse, productive, and confident contributors as well as a proliferation of best practices and growth. To evaluate whether these expectations correspond to reality, the authors quantitatively analyzed 84 months of historical code changes, issue reports, and code reviews. Through this study, the authors confirmed that 65% more women and nonbinary developers have participated in the onboarding program as opposed to those who chose not to participate. Moreover, developers who participated in the onboarding program are more versatile in their technical skills and are more likely to be hired. Onboarding increases the contributors’ productivity and retention rate, reducing both the effort required from the contributors to get their work accepted and the number of bugs in their contributions. As such, the onboarding program at OpenStack can be seen as highly successful and might encourage other ecosystems and companies to invest in onboarding. This paper appears in the main research track of ICSE 2021. Access it at http://bit.ly/PD-2021-Sept-03.

Extraction Rationale for Development Decisions

“Extracting Rationale for Open Source Software Development Decisions—A Study of Python Email Archives,” by Pankajeshwara Sharma and colleagues, investigates the hidden rationale behind the decisions that lead to the maintenance and evolution of software products and how to extract this rationale to promote transparency and accountability. The paper presents Rationale Miner, a tool that uses heuristics for rationale extraction in open source software projects. To extract the rationale, the authors analyze potential decisions captured in the Python email archives stored in GitHub and from 248 Python Enhancement Proposals (PEPs) that include their states (accepted and rejected). Rationale Miner uses several heuristics to analyze thousands of emails and extract the ground truth and patterns for statements of rationale. From the total data set containing around 1.5 million email messages, the authors manually
Onboarding increases the contributors’ productivity and retention rate, reducing both the effort required from the contributors to get their work accepted and the number of bugs in their contribution.