Beyond Code: Towards Intelligent Collaboration Tools

Vladimir Kovalenko[1]

[1]JetBrains Research, JetBrains N.V., Huidekoperstraat 26, 1017 ZM Amsterdam, Noord-Holland, The Netherlands

Abstract

Think of a software engineer at work. What is on their screen: a terminal window? An IDE? In practice, it is just as likely to be a messenger or a bug tracker. We have made impressive progress with enabling individual developer tools to boost productivity through smart and efficient code analysis. Refactor a large project? A few keystrokes will do. Explore the structure of a complex system? No problem, click here, hope your screen is big enough. IDEs are incredibly powerful. In contrast, the collaboration tools of today — think issue trackers, code review tools, messenger workspaces — still resemble bulletin board systems. Most of their beauty and complexity lies in reliability, performance, and UX, rather than in features that truly model, support, and enhance the process of collaborative work. While there is plenty of room for new data-driven approaches in real-world collaboration tools, these tools are less popular as a context for such approaches proposed by the research community. In the keynote talk[1], I am looking to highlight the collaboration tools as particularly interesting targets for data-driven enhancement.

Keywords

Intelligent Collaboration Tools, Data-Driven Software Engineering

1. "Software engineers mostly code"

Contrary to popular belief, this is a questionable statement. While coding is one of the primary activities of engineers, studies reveal that activities involving editing code, in fact, only occupy about a half of the working hours for those working in teams, while most of the remaining time is dedicated to collaborative activities such as code review, communication, planning, and task tracking [1, 2, 3]. However, popular culture, including movies and stock images, mostly presents a developer as a person in front of a computer with code on their screen.

Despite extensive research around the activities of software engineers, this code-centric bias is present in the software engineering research community as well. Much of the research presented in major research venues proposes new techniques that could potentially enhance existing software engineering tools (e.g. bug prediction [4]), or improving the techniques already present in modern tools (e.g. code completion [5]). In the recent years, the majority of such practice-oriented research has been aimed at facilitating code manipulation and maintenance, rather than collaborative activities.

[1]Slides: https://vovak.me/assets/benevol-21-keynote.pdf

BENEVOL'21: The 20th Belgium-Netherlands Software Evolution Workshop, December 07–08, 2021, 's-Hertogenbosch (virtual), NL

vladimir.kovalenko@jetbrains.com (V. Kovalenko)
https://vovak.me/ (V. Kovalenko)
0000-0001-5880-7323 (V. Kovalenko)

2. Intelligent Collaboration Tools

Unlike integrated development environments (IDEs) that offer incredibly powerful and complex features enabling manipulation, refactoring, and analysis of large software projects with minimal input from the user, and thus saving their time and energy, the collaboration tools of today – issue trackers, code review systems, messenger workspaces, etc. – are relatively simple, and do not extensively model the processes they support or offer many "smart" features to their users. While some approaches, such as expert recommendation [6], have found their way into mainstream collaboration tools such as Github and Gerrit, these examples are still rather rare.

The room for improvement of the tools, along with the bias towards coding activities (Section 1), call for action to improve the collaboration tools. The list below presents some of the promising research directions.

- Gaining a better understanding of users' behaviour, issues, and needs in collaboration tools. This way, we can maximize the value of new techniques and features for end users.
- Trying academic approaches to data-driven support in collaborative engineering in practice by extending existing tools.
- Devising techniques to ensure long-term health of projects and team dynamics.
- Treating the process and data in collaboration tools as an artifact: enabling the tools to highlight potentially inefficient process patterns and suggest improvements;
- Augmenting the tools with extensive analytics engines to help their users comprehend and analyze complex systems and processes.

At the Intelligent Collaboration Tools Lab (ICTL),[1] we work in these and related directions. We are open to collaboration.

References

[1] J. Singer, T. Lethbridge, N. Vinson, N. Anquetil, An examination of software engineering work practices, in: CASCON First Decade High Impact Papers, 2010, pp. 174–188.

[2] A. N. Meyer, T. Fritz, G. C. Murphy, T. Zimmermann, Software developers' perceptions of productivity, in: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 19–29.

[3] M. K. Gonçalves, C. R. de Souza, V. M. González, Collaboration, information seeking and communication: An observational study of software developers' work practices., J. Univers. Comput. Sci. 17 (2011) 1913–1930.

[4] M. D'Ambros, M. Lanza, R. Robbes, An extensive comparison of bug prediction approaches, in: 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE, 2010, pp. 31–41.

[5] A. Svyatkovskiy, S. Lee, A. Hadjitofi, M. Riechert, J. V. Franco, M. Allamanis, Fast and memory-efficient neural code completion, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), IEEE, 2021, pp. 329–340.

[6] H. A. Çetin, E. Doğan, E. Tüzün, A review of code reviewer recommendation studies: Challenges and future directions, Science of Computer Programming (2021) 102652.

[1] https://research.jetbrains.org/groups/ictl