The Social Developer: The Future of Software Development

Tom Mens, University of Mons

Marcelo Cataldo, Uber Advanced Technologies Group

Daniela Damian, University of Victoria

Digital Object Identifier 10.1109/MS.2018.2874316
Date of publication: 8 January 2019

CONTEMPORARY SOFTWARE ENGINEERING has inevitably become much more social. Due to the size, complexity, and diversity of today's software systems, there is a need to interact across organizational, geographical, cultural, and socioeconomic boundaries. Large-scale software development now implies active user involvement and requires close cooperation and collaboration between team members and all types of development activities. Members of software projects across all roles must communicate and interact continuously with other project members as well as with a variety of stakeholders, such as users, analysts, suppliers, customers, and business partners. This theme issue aims to inform software engineering practitioners about current trends and recent advances in research and practice of sociotechnical analysis and support for large-scale software development.

Mirroring Hypothesis

The idea that the structure of a system is in some way related to the structure of the organization building it has been studied for decades across many industries.[9] In software systems, Melvin Conway argued that the structure of a system mirrors the structure of the organization that designed it, as an inevitable consequence of the communication needs of the people involved in developing the system.[2] Twenty years ago, Herbsleb and Grinter studied Conway's law specifically in the context of geographically distributed development teams and found the lack of sociotechnical congruence to be a major driver of project failure.[6] This observation has become even more relevant today, with the omnipresence of open source and mixed-source software development as well as development projects that need to adapt to fast-changing environments.[9]

Social Coding

Interestingly, a popular myth tells that software developers act as lone wolfs, creating software systems in isolation without much interaction with the outside world. In today's world of complex software systems, the inverse is true. The need for coordination and collaboration has led, among other factors, to the phenomenon of *social coding*, in which team members rely on integrated platforms and collaborative authoring tools for enabling and improving their communication and interaction. In that context, software project members rely on tools like wikis, bug and issue trackers, discussion forums, code repositories, Q&A websites, mailing lists, commenting and reviewing mechanisms, and many more. During the past decade, researchers have studied social coding platforms and have found, for instance, that issue trackers have become essential collaboration instruments;[1] social coding practices can evolve into effective strategies for coordinating work;[4] and through mechanisms that leverage and enhance the inherent transparency, social coding platforms can become very valuable collaborative environments for large projects.[6]

When embarking into developing such large-scale software systems, the social aspects and their relationship to the technical dimension of projects cannot be underestimated. This theme issue focuses on the current trends and advances in the research and the practice related to the social and sociotechnical aspects of software development. We aim to provide useful guidelines and recommendations to software developers on how to embrace the social side of software development.

Social Factors and the Evolving Landscape of Software Projects

The understanding of the role of social factors in software development projects has evolved significantly during past decades. The trend toward geographically distributed development in the 1990s drew attention to communication and coordination problems. In the late 1990s and early 2000s, cultural diversity and the various dimensions of geographic dispersion as well as the structure of collaboration patterns also started to surface as critical forms of social factors. The world continues to evolve, and today we are faced with projects that operate in an open source environment, in traditional corporate settings, in corporate settings but following open source or community-based approaches, and any possible blend of those various approaches.

Mixing open source and proprietary software strategies is increasingly seen as beneficial by software-producing companies. By combining the best of both worlds, companies could benefit from the advantages of both business models. However, significant disconnects may occur: the open source community may be resistant to company involvement; the company may impose its strategic vision too much on the developer community; there may be incompatible work practices and processes leading to communication and collaboration problems. A well-known illustration of what such problems can lead to was the fork of the open source OpenOffice software suite into LibreOffice in 2011. When Sun Microsystems (who owned OpenOffice) was taken over by Oracle Corporation, it led to a reduced openness of OpenOffice, resulting in the LibreOffice fork. This fork is still very much alive today and is perceived by its community as supportive, diversified, and independent.[5]

As projects become larger and open source and proprietary software strategies blend together, the social aspects of diversity and inclusion are becoming increasingly important. Large communities can benefit from a high diversity of contributors, regardless of the dimension of diversity considered (e.g., cultural diversity, gender diversity, seniority, inclusion of people with disabilities, and positive discrimination to address underrepresentation of minority groups). Vasilescu et al.[7] studied the effect of gender and tenure diversity and found them to be positive and significant predictors of productivity of open source development teams. Codes of conduct are becoming commonplace in major software development communities and distributed platforms. As a recent example, Stack Overflow updated its code of conduct in August 2018 to encourage kindness and constructive feedback and to enforce mutual respect and repress unacceptable behavior and other kinds of misconduct.

All of this clearly illustrates the importance of the social phenomenon in large-scale software development, motivating the need and timeliness of this special issue.

In This Issue

We received 21 submissions for this theme issue. Of these, eight were invited to prepare a revised version, based on their relevance to software practitioners. Five of those were selected for inclusion in the theme issue. They cover a wide variety of topics related to sociotechnical factors of software development. It should not come as a surprise that many of these studies focus on the software ecosystem point of view, because it is at this level that contributors of different but interrelated software projects need to interact and collaborate, making it more likely for sociotechnical problems to emerge.

In "Designing Corporate Hackathons with a Purpose," Than et al. focus on the use of corporate hackathons. Such hackathons are traditionally seen as a way of addressing technical challenges and achieving such business needs as product innovation in relatively short periods of time. In addition to this, they can also be a very efficient tool to achieve a variety of social goals, such as enriching intracompany social networks, facilitating collaborative learning, and preparing employees for future changes and positions. If designed carefully, hackathons can achieve several of these goals.

In "Self-Managing: An Empirical Study of the Practice in Agile Teams," Gutiérrez et al. investigate the value of the agile technique of self-management in software development teams. Through a survey conducted with 247 mostly Hispanic subjects across 22 countries, they reveal that teams with a high perception of autonomy perceive a high level of self-management through their leadership styles and the language used to describe their tasks. This study may help practitioners in diagnosing the degree of self-management in their own organizations and carrying out the necessary steps to further increase this self-management, if desired.

In the "OpenStack Gender Diversity Report," Izquierdo et al. focus on the sociotechnical aspects of diversity and inclusion of underrepresented minorities by analyzing the current state of gender diversity within the open source community of OpenStack contributors. The analyzed data include both code and noncode contributions. The importance of this article lies in the awareness of the need to embrace diversity as well as the way in which such diversity can be measured and promoted. Initiatives like Linux Foundation's CHAOSS community will help to achieve these goals.

In "Toward Solving Social and Technical Problems in Open Source Software Ecosystem," Marsan et al. carried out a cause-and-effect analysis for identifying typical problems during large-scale distributed development of open source software systems. Based on in-depth interviews with 10 contributors to open source ecosystems, they found loss of contributors to be one of the most important social problems and poor code quality to be one of the major technical problems, with both problems resulting from complex sociotechnical interrelations of causes.

In "What Characterizes an Influencer in Software Ecosystems?" Farias et al. focus on JavaScript's npm package management ecosystem. The authors conducted a survey with 242 developers who contributed to GitHub repositories for npm packages. By doing so, they gather insights on how developers influence the ecosystem in which they take part and turn this into actionable advice for ecosystem managers, integrators, and collaborators.

Acknowledgments
This work is partially financed by the Fund for Scientific Research–Fonds de recherche du Québec–Wallonia collaborative research project "SECOHealth: Towards an Interdisciplinary, Socio-Technical Methodology and Analysis of the Health of Software Ecosystems."

ABOUT THE AUTHORS

TOM MENS is a full professor at the University of Mons, Belgium, with the Department of Computer Science. His research interests include software evolution, software modeling, software ecosystems, open source software, and empirical software engineering. Mens received a Ph.D. in computer science from the Vrije Universiteit Brussel, Belgium. He is a Senior Member of the IEEE and a member of the IEEE Computer Society. Contact him at tom.mens@umons.ac.be.

MARCELO CATALDO is a software engineering manager with the Uber Advanced Technology Group. His research interests include geographically distributed software development with a special focus on the relationship between the software architecture and the organizational structure in large-scale software development projects. Cataldo received a Ph.D. in societal computing from Carnegie Mellon University. Contact him at chelo.cataldo@gmail.com.

DANIELA DAMIAN is a professor of software engineering at the University of Victoria with the Department of Computer Science. Her research interests include software engineering, requirements engineering, computer-supported cooperative work, and empirical software engineering. Damian obtained her Ph.D. in software engineering from the University of Calgary, Canada. She is a Member of the IEEE. Contact her at damian.daniela@gmail.com.

References

1. D. Bertram, A. Voida, S. Greenberg, and R. Walker, "Communication, collaboration, and bugs: The social nature of issue tracking in small, collocated teams," in *Proc. ACM 2010 Conf. Computer Supported Collaborative Work*, pp. 291–300.
2. M. E. Conway, "How do committees invent?" *Datamation*, vol. 14, no. 4, pp. 23–28, 1968.
3. K. Crowston. (2005). The social structure of free and open source software development. *First Monday*. [Online]. *10(2)*. Available: http://www.firstmonday.dk/ojs/index.php/fm/article/view/1207/1127
4. L. Dabbish, C. Stuart, J. Tsay, and J. D. Herbsleb, "Social coding in GitHub: Transparency and collaboration in an open software repository," in *Proc. ACM 2012 Conf. Computer Supported Cooperative Work*, pp. 1277–1286.
5. J. Gamalielsson and B. Lundell, "Sustainability of open source software communities beyond a fork: How and why has the LibreOffice project evolved?" *J. Syst. Softw.*, vol. 89, pp. 128–145, Mar. 2014.
6. J. D. Herbsleb and R. E. Grinter, "Architectures, coordination, and distance: Conway's law and beyond," *IEEE Softw.*, vol. 16, no. 5, pp. 63–70, 1999.
7. B. Vasilescu, D. Posnett, B. Ray, M. G. J. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov, "Gender and tenure diversity in GitHub teams," in *Proc. ACM Conf. Human Factors in Computing Systems (CHI)*, 2015 , pp. 3789–3798.
8. S. Datta, "How does developer interaction relate to software quality? An examination of product development data," *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1153–1187, 2018.
9. L. J. Colfer and C. Y. Baldwin, "The mirroring hypothesis: Theory, evidence, and exceptions," *Ind. Corporate Change*, vol. 25, no. 5, pp. 709–738, 2016.