Towards a Data-Driven Understanding of Cross-team Collaboration

Han Yong Wunrow
Sandia National Laboratories
nhwunro@sandia.gov

Reed Milewicz
Sandia National Laboratories
rmilewi@sandia.gov

Elaine M. Raybourn
Sandia National Laboratories
emraybo@sandia.gov

***Index Terms*—scientific software teams; developer networks**

I. Introduction

Scientific software is increasingly critical to the practice of science, and as the demand for that software has grown, so too has the need for more effective teaming and collaboration. While historically most scientific software has been the product of individuals and small teams, scientific software development today is increasingly large-scale, open-source, community-driven, multi-disciplinary, and multi-institutional.

The question of how research teams function and how we can empower them represents an exciting frontier for software engineering research. Scientific software developer teams would benefit from better and more tailored tools, techniques, and methodologies to enable more effective collaboration. However, as Dennehy and Conboy observe, the culture and context of a software project are "critical determinants of software development success" and that "a method, practice, or tool cannot be studied in isolation" [1]. That is, through a better understanding of current teaming and collaboration practices, we can arrive at better ways of working and – by extension – do better science.

In this paper, we explore how data-driven modeling of large-scale scientific software collaborations can inform the broader conversation on what research teams need to be successful. Ongoing work on software repository mining and network science techniques provide us with the necessary tools to extract and analyze team dynamics [2], [3].

II. Background

Software engineering research, also known as software science, is the study of software systems and their development, operation, and maintenance. As a branch of computer science, software science applies a sociotechnical lens to help model, understand, and predict the factors that lead to high-quality software systems. In recent decades, the move towards open-source, community software has enabled software scientists to rigorously study software development practices at scale. By combining software repository mining techniques with insights from data science, network science, and machine learning, researchers can derive actionable insights regarding software development team practices.

While the social networks and collaboration habits of scientists have been the subject of decades of research [4], scientific software developers have long been understudied in the context of empirical software engineering. This is because, historically, scientific software development has often been a private affair. However, with the shift towards open science and community software hosted on open-source platforms, we now have access to rich sources of fine-grained data on scientific software development; questions about team practices and collaboration behaviors are finally amenable to data-driven analysis.

III. Network Science

Network science provides a logical framework to model the interactions between teams, where nodes represent contributors to a project and edges represent some relationship between contributors.

However, to build a rigorous foundation for research in this space, we must carefully define what we mean by a scientific software team and its contributors. Thompson et al. define a team as "a group of people who are interdependent with respect to information, resources, knowledge and skills and who seek to combine their efforts to achieve a common goal" [5].

Network science also allows us to analyze and test hypotheses about the interdependencies and dynamics of teams. Previous studies of open-source software projects have used social networks to characterize the development process and describe how the structure of these networks can be used to effectively manage software development projects [6]–[8]. For example, Roach et al. examined the open-source software development of the Python programming language and found that network measures proved to be an accurate measure of contribution by an individual developer within a community.

However, in the context of teams who build and use scientific software, Ramin et al. define contribution as "any activity, demanding human resources, that adds to the fulfillment of project goals, by adding value to the developed product or the (future) effectiveness of the team" [9]. This definition necessitates the inclusion of roles beyond developers when analyzing cross-team collaboration.

Therefore, when constructing a contributor social network, it is important to consider our precise definition of a contributor. If we form a network solely on changes to repositories (e.g., commits, pull requests, merge requests, patches), we may miss key contributors such as project managers and principal investigators who may not necessarily actively contribute to the

codebase. Young et al. argue that code as a measure for contribution likely does not capture all contributors and provide four alternative models of contributor acknowledgment [10]. Additionally, it is important to consider the temporal aspect of interactions within networks. While two people may have interacted with the same repository, it may have been several years apart. In this case, it becomes useful to communicate with members of software teams to obtain the ground truth of who is a contributor, the degree to which they contribute, and when the contributions take place.

The software engineering research community has acknowledged that source lines of code, number of bugs fixed, number of tasks completed, and hours worked are poor measures of productivity and contribution value. This is because quantity solely does not imply software quality or the complexity of the code [8]. Relevant metrics of the importance of a contribution could be used instead when defining the edge relationship and weight in the network.

A recent study showed that analyzing community structure is the dominant research direction in studies on developer social networks [11]. Generally, communities are groups of nodes with a higher probability of being connected to each other than nodes in other groups [12]. Understanding the evolution of communities in networks gives us a picture of which contributors and teams interact with whom and to what degree.

IV. Discussion

While data-driven approaches promote understanding of how teams function, basing decisions solely on data is misguided. One major concern is the quality of data being used. There are certainly methods to reflect the degree of uncertainty based on the quality of the data; however, teams cannot accurately measure with poor-quality data. Additionally, it is best to keep some healthy skepticism when interpreting results from data-driven approaches. Before accepting claims, it is best to consult with a subject-matter expert on the particular topic being investigated. In particular, research on teams would benefit from the expertise of social scientists who have been researching teams for decades. Taking these limitations into account is crucial when interpreting and acting on results from data-driven approaches.

There is a growing body of evidence that shows collaborative teams tend to produce the highest-impact scientific work [13]. This claim motivates our research objective of understanding how teams interact with each other. Empirically studying scientific software team dynamics enables us to examine current teaming and collaboration patterns and then develop practices to sustain the health and success of teams. Specifically, network science methods allow us to identify key contributors within the network, detect community structures as projects progress, and uncover complex dependencies between teams. A network science approach also focuses less on individual aspects of the project and instead measures the entire ecosystem in which teams function together, making this research more generalizable and actionable.

V. Conclusion

Leveraging a data-driven approach to analyze scientific software team ecosystems will help advance the field's understanding of large-scale scientific software collaborations and help develop ways to promote more productive and sustainable software development. Network science is a well-suited framework for modeling cross-team collaboration and will help answer questions on how to best structure teams together to facilitate collaboration, where key leverage points within networks of teams are, and what kind of teaming and collaboration practices lead to increased productivity and risk mitigation.

VI. Acknowledgements

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DENA0003525. **SAND2021-7356 C**.

References

[1] D. Dennehy and K. Conboy, "Going with the flow: An activity theory analysis of flow techniques in software development," *Journal of Systems and Software*, 2016.

[2] A. Santos, M. Souza, J. Oliveira, and E. Figueiredo, "Mining software repositories to identify library experts," *ACM International Conference Proceeding Series*, no. i, pp. 83–91, 2018.

[3] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community Structure in Time-Dependent, Multiscale, and Multiplex Networks," *Science*, vol. 328, no. 5980, pp. 876–878, may 2010. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0219525903001067 https://www.sciencemag.org/lookup/doi/10.1126/science.1184819

[4] C. Roth, "Socio-semantic frameworks," *Advances in Complex Systems*, vol. 16, no. 04n05, p. 1350013, 2013.

[5] L. L. Thompson and M. Thompson, "Making the team: A guide for managers," 2008.

[6] S. Kumar, "Using social network analysis to inform management of open source software development," *Proceedings of the Annual Hawaii International Conference on System Sciences*, vol. 2015-March, pp. 5154–5163, 2015.

[7] G. Zanella and C. Z. Liu, "A Social Network Perspective on the Success of Open Source Software: The Case of R Packages," *Proceedings of the 53rd Hawaii International Conference on System Sciences*, vol. 3, pp. 471–480, 2020.

[8] C. Roach and R. Menezes, "Using networks to understand the dynamics of software development," *Communications in Computer and Information Science*, vol. 116 CCIS, no. January, pp. 119–129, 2011.

[9] F. Ramin, C. Matthies, and R. Teusner, "More than Code: Contributions in Scrum Software Engineering Teams," *Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020*, pp. 137–140, 2020.

[10] J.-G. Young, A. Casari, K. McLaughlin, M. Z. Trujillo, L. Hébert-Dufresne, and J. P. Bagrow, "Which contributions count? Analysis of attribution in open source," 2021. [Online]. Available: http://arxiv.org/abs/2103.11007

[11] S. Herbold, A. Amirfallah, F. Trautsch, and J. Grabowski, "A systematic mapping study of developer social network research," *Journal of Systems and Software*, vol. 171, p. 110802, 2021. [Online]. Available: https://doi.org/10.1016/j.jss.2020.110802

[12] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics Reports*, vol. 659, pp. 1–44, 2016.

[13] R. R. Schreiber and M. P. Zylka, "Social Network Analysis in Software Development Projects: A Systematic Literature Review," pp. 321–362, mar 2020.