Sustaining Scientific Open-Source Software Ecosystems: Challenges, Practices, and Opportunities

Jiayi Sun
University of Toronto

ABSTRACT

Scientific open-source software (scientific OSS) has facilitated scientific research due to its transparent and collaborative nature. The sustainability of such software is becoming crucial given its pivotal role in scientific endeavors. While past research has proposed strategies for the sustainability of the scientific software or general OSS communities in isolation, it remains unclear when the two scenarios are merged if these approaches are directly applicable to developing scientific OSS. In this research, we propose to investigate the unique challenges in sustaining the scientific OSS ecosystems. We first conduct a case study to empirically understand the interdisciplinary team's collaboration in scientific OSS ecosystems and identify the collaboration challenges. Further, to generalize our findings, we plan to conduct a large-scale quantitative study in broader scientific OSS ecosystems to identify the cross-project collaboration inefficiencies. Finally, we would like to design and develop interventions to mitigate the problems identified.

ACM Reference Format:
Jiayi Sun. 2024. Sustaining Scientific Open-Source Software Ecosystems: Challenges, Practices, and Opportunities. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3639478.3639805

1 INTRODUCTION

Scientific software, the software or underlying computing infrastructure used in scientific domains, such as chemistry, biology, physics, and astronomy, has become a critical component of modern scientific research [10]. The sustainability of scientific software, defined as "the ability to maintain the software in a state where scientists can understand, replicate, and extend prior reported results that depend on that software" [53] is increasingly vital for stakeholders involved, given the crucial role of scientific software in the scientific process.

However, developing and maintaining scientific software is a non-trivial task due to the complexity of the scientific domain, where scientists possess the necessary scientific expertise but often lack adequate software engineering training for ensuring software quality [12, 33, 39, 40, 50]. Therefore, ***interdisciplinary teams*** consisting of (but not limited to) scientists and software development engineers (SDEs) need to work together to develop scientific software [14, 35, 44]. These groups have different goals, training, and experience, often leading to friction in the process. Evidence shows that tension often arises between scientists and SDEs when prioritizing the focus and workload of the project [27, 28, 38, 44].

With the success of the ***open-source model***, the development of scientific software in the open-source environment has presented great potential to benefit scientific discoveries (e.g., Numpy [6] for array programming). Multiple software projects are often used in research workflow to prevent duplicated efforts and improve working efficiency. The emergence of scientific OSS ecosystems, consisting of multiple software projects offering different functionalities in similar domain contexts, such as the ImageJ ecosystem for scientific image analysis [43] and the Bioconductor ecosystem for biology research [20], have facilitated scientific processes in the corresponding domains by promoting the reuse of the existing software and open collaboration. However, the open-source development model also suffers from sustainability concerns, in particular from the community's perspective, such as difficulties in retaining contributors and attracting newcomers, due to its voluntary contribution nature [16]. The combination of *interdisciplinary collaboration* and *open-source mechanisms*, despite its benefits, raises questions about the relevance of existing solutions designed for individual challenges within each context.

1.1 The Problem

Sustainable Collaboration. Prior studies investigated issues of interdisciplinary collaboration in developing scientific software and proposed solutions like increased education [54] and hackathons to facilitate knowledge exchange [39]. However, it's uncertain if these solutions apply to open-source settings that are characterized by remote, asynchronous collaboration among diverse groups who mostly are making voluntary contributions. Additionally, while interdisciplinary collaboration challenges between SDEs and data scientists [36], and between SDEs and user experience (UX) designers [9, 32] have been studied, the collaboration challenges between scientists and SDEs in the scientific OSS setting remain underexplored, and it is necessary to understand and identify the unique challenges when developing scientific OSS.

Sustainable Community. Researchers have investigated the sustainability challenges of general OSS from different perspectives, such as the motivations of contributors [21], the challenges for newcomers [16, 24, 48, 49], and the burnout of existing contributors [42]. Corresponding best practices, such as mentoring [17], Good First Issues [51], and summer of code programs [47] are adopted to mitigate the problems. However, scientific OSS differs in funding resources and stakeholder composition [7], potentially posing distinct challenges. *Therefore, the applicability of existing*

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0502-1/24/04
https://doi.org/10.1145/3639478.3639805

solutions, such as effective governance and sponsorships [19, 46], to the unique context of scientific OSS communities remains questionable.

Sustainable Ecosystem. As the scientific OSS ecosystem consists of multiple projects, the sustainability concerns would also escalate to the ecosystem level as coordination and collaboration among multiple projects are required. Prior studies have explored OSS ecosystems with a main focus on the code dependency relationship (e.g., cross-project bug fixing [13, 15, 30], breaking changes [11], software supply chain [29, 52]), and the practices of cross-project code reuse [22]. Additionally, various methods are designed to improve coordination efficiency such as impact analysis of cross-project bugs on downstream modules [31], social network analysis to illustrate the relationship among developers and projects to support knowledge collaboration across projects [37], and dependency management tools to improve the quality of the dependency network [26]. *However, given that scientific software primarily consists of individual or group-developed components tailored to specific research needs and not necessarily interdependent, it is unclear whether the existing solutions proposed for general OSS ecosystems focusing on dependency-relationship (e.g., NPM ecosystem [5]) remain effective for cross-project collaboration within scientific OSS ecosystems.*

Such problems have been also recognized by various organizations and funding bodies. For example, Chan Zuckerberg Initiatives (CZI) [4] and Alfred P.Sloan Foundation [3] have dedicated grants to support the maintenance and improve the sustainability of the scientific OSS ecosystems.

1.2 Related Work on Scientific OSS

Scientific OSS communities published papers to demonstrate feature designs, discuss challenges, as well as share experiences on the efforts to sustain the community [25, 41]. Milewicz et al. studied seven scientific OSS software and the corresponding software teams. They grouped the team members based on their levels of seniority and found that senior research staff (e.g. professors) are responsible for half or more of commits, juniors (e.g. graduate students) also contribute substantially, and third-party contributors are scarce [34]. Additionally, Sharma et al. developed a model to automatically detect different types of technical debts in the development process of R packages and empirically identified the causes of the technical debts [45]. Little has been studied regarding the collaboration and sustainability challenges in scientific OSS ecosystems. Different from prior work, this research aims to investigate the unique challenges in developing sustainable scientific OSS ecosystems, as well as identify the opportunities for designing tooling support to address the challenges.

1.3 Research Questions

We hypothesize that the combination of scientific software and open-source mechanisms would introduce distinct challenges in developing and maintaining sustainable scientific OSS ecosystems. Therefore, we ask the following research questions: (1) **RQ1:** What are the unique challenges for interdisciplinary teams in developing sustainable scientific OSS? (2) **RQ2:** How do contributors collaborate across projects in the scientific OSS ecosystem? (3) **RQ3:** What are the opportunities for designing and developing interventions to improve the sustainability of the scientific OSS ecosystem?

2 RESEARCH PLAN

Objective-1: To ***understand the challenges in developing scientific OSS (RQ1)***, we conducted a case study with the Astropy ecosystem [8], a popular scientific OSS ecosystem in the astronomy domain of which the core package [1] has over 1.7k forks on GitHub. We applied a mixed-method approach [18, 23], including interviews with core contributors, a survey with disengaged contributors, and mining the development artifacts, such as source code, issue discussions, and pull requests in the repositories of the Astropy ecosystem that are hosted on GitHub.

Preliminary Results. From the case study, we observe the tensions in the interdisciplinary team collaboration regarding (1) development tasks prioritization and (2) the perception of seniority of contributors on the team. Moreover, we find out that the motivations for contributing to scientific OSS ecosystems are mostly because of the need for scientists' own research. Meanwhile, the top reason for disengagement is the career focus shift (e.g., research topic change). We also identified inefficiencies during collaboration such as duplicate code, fragmented implementation, wasted effort, and lack of awareness within the ecosystem.

Objective-2: To ***understand the intentions of cross-project collaboration in the scientific OSS ecosystem and identify corresponding inefficiencies (RQ2)***, we plan to conduct a large-scale quantitative study in broader scientific OSS communities to verify and generalize the findings from the case study described before. We plan to leverage the cross-reference mechanism between issue discussions on GitHub [2] to approximate cross-project communication and collaboration. Through constructing the cross-reference graphs with the cross-project communication links and the corresponding issue discussions, we will analyze and identify the intentions of cross-project collaboration. corresponding inefficiencies, and describe the existing practices.

Objective-3: To ***design new interventions to better support sustainable scientific OSS development (RQ3)***, we plan to combine the insights identified in the previous objectives to design and develop tooling support and/or best practices. Potential solutions include but not limited to GitHub bots to assist maintainers and contributors to better organize the artifacts and share knowledge. Further, we will design user experiments to evaluate the effectiveness and usefulness of the interventions. Moreover, we will also reach out to broader scientific OSS practitioners to validate the findings, collect feedback, and enhance the impact of our work.

3 CONTRIBUTION AND POTENTIAL IMPACT

With the research objectives achieved, we will contribute to the body of knowledge in the following aspects: (1) A better understanding of the problem space in developing sustainable scientific OSS ecosystems. We hope to offer practical advice for developers in various scientific software communities. (2) The knowledge of the intentions, practices, and inefficiencies of cross-project collaboration, will contribute to improving the sustainability of scientific OSS on the ecosystem level. (3) The interventions designed will contribute to the design and development of techniques and best practices to further ensure the sustainability and efficiency of developing the scientific OSS ecosystem.

REFERENCES

[1] 2023. Astropy: core library. https://github.com/astropy/astropy
[2] 2023. Autolinked references between issues and PRs. https://shorturl.at/hjEKQ
[3] 2023. Better Software for Science. https://sloan.org/programs/digital-technology/better-software-for-science
[4] 2023. Essential Open Source Software for Science. https://chanzuckerberg.com/rfa/essential-open-source-software-for-science/
[5] 2023. npm, Inc. https://www.npmjs.com/about
[6] 2023. *NumPy*. https://numpy.org/
[7] 2023. Overview of research software funding landscape. https://researchsoft.org/blog/2022-02-24/
[8] 2023. *The Astropy Project*. https://www.astropy.org/
[9] Ohoud Almughram and Sultan Alyahya. 2017. Coordination support for integrating user centered design in distributed agile projects. In *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*. IEEE, 229–238.
[10] Elvira-Maria Arvanitou et al. 2021. Software engineering practices for scientific software development: A systematic mapping study. *Journal of Systems and Software* 172 (Feb. 2021), 110848.
[11] Chris Bogart et al. 2021. When and How to Make Breaking Changes: Policies and Practices in 18 Open Source Software Ecosystems. *ACM Transactions on Software Engineering Methodology* 30, 4 (2021).
[12] Bozhidar Bozhanov. 2014. The Low Quality of Scientific Code. https://techblog.bozho.net/the-astonishingly-low-quality-of-scientific-code/
[13] Gerardo Canfora et al. 2011. Social interactions around cross-system bug fixings: the case of FreeBSD and OpenBSD. In *Proc. Working Conf. Mining Software Repositories (MSR)*. 143–152.
[14] Jeffrey C Carver et al. 2007. Software development environments for scientific and engineering software: A series of case studies. In *29th International Conference on Software Engineering (ICSE'07)*. Ieee, 550–559.
[15] Zhifei Chen et al. 2022. Collaboration in software ecosystems: A study of work groups in open environment. *Information and Software Technology* 145 (2022), 106849.
[16] InduShobha Chengalur-Smith et al. 2010. Sustainability of free/libre open source projects: A longitudinal study. *Journal of the Association for Information Systems* 11, 11 (2010), 5.
[17] Fabian Fagerholm et al. 2014. The role of mentoring and project characteristics for onboarding in open source software projects. In *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement*. 1–10.
[18] Bent Flyvbjerg. 2006. Five misunderstandings about case-study research. *Qualitative inquiry* 12, 2 (2006), 219–245.
[19] Jonas Gamalielsson and Björn Lundell. 2014. Sustainability of Open Source software communities beyond a fork: How and why has the LibreOffice project evolved? *Journal of Systems and Software* 89 (2014), 128–145.
[20] Robert C Gentleman et al. 2004. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology* 5, 10 (2004), 1–16.
[21] Marco Gerosa et al. 2021. The Shifting Sands of Motivation: Revisiting What Drives Contributors in Open Source. In *Proc. Int'l Conf. Software Engineering (ICSE)*. 1046–1058.
[22] Mohammad Gharehyazie et al. 2017. Some from here, some from there: Cross-project code reuse in github. In *Proc. Working Conf. Mining Software Repositories (MSR)*. IEEE, 291–301.
[23] Timothy C Guetterman and Michael D Fetters. 2018. Two methodological approaches to the integration of mixed methods and case study designs: A systematic review. *American Behavioral Scientist* 62, 7 (2018), 900–918.
[24] Christoph Hannebauer and Volker Gruhn. 2017. On the relationship between newcomer motivations and contribution barriers in open source projects. In *Proceedings of the 13th International Symposium on Open Collaboration*. 1–10.
[25] Charles R Harris et al. 2020. Array programming with NumPy. *Nature* 585, 7825 (2020), 357–362.
[26] Joseph Hejderup et al. 2018. Software ecosystem call graph for dependency management. In *Proc. Int'l Conf. Software Reuse (ICSR)*. 101–104.
[27] James Howison and James D Herbsleb. 2011. Scientific software production: incentives and collaboration. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*. 513–522.
[28] Diane Kelly. 2015. Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software. *Journal of Systems and Software* 109 (2015), 50–61.
[29] Riivo Kikas et al. 2017. Structure and evolution of package dependency networks. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 102–112.
[30] Wanwangying Ma et al. 2017. How Do Developers Fix Cross-Project Correlated Bugs? A Case Study on the GitHub Scientific Python Ecosystem. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. 381–392.
[31] Wanwangying Ma et al. 2020. Impact analysis of cross-project bugs on software ecosystems. In *Proc. Int'l Conf. Software Engineering (ICSE)*. 100–111.
[32] Nolwenn Maudet et al. 2017. Design breakdowns: designer-developer gaps in representing and interpreting interactive systems. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. 630–641.
[33] Zeeya Merali. 2010. Computational science:... error. *Nature* 467, 7317 (2010), 775–777.
[34] Reed Milewicz, Gustavo Pinto, and Paige Rodeghero. 2019. Characterizing the roles of contributors in open-source scientific software projects. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 421–432.
[35] Chris Morris and Judith Segal. 2009. Some challenges facing scientific software developers: The case of molecular biology. In *2009 Fifth IEEE International Conference on e-Science*. IEEE, 216–222.
[36] Nadia Nahar et al. 2022. Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process. *Organization* 1, 2 (2022), 3.
[37] Masao Ohira et al. 2005. Accelerating cross-project knowledge collaboration using collaborative filtering and social networks. In *Proceedings of the 2005 international workshop on Mining software repositories*. 1–5.
[38] Drew Paine and Charlotte P. Lee. 2017. "Who has plots?": Contextualizing scientific software, practice, and visualizations. *Proceedings of the ACM on Human-Computer Interaction* 1 (2017), 1–21. Issue CSCW.
[39] Ei Pa Pa Pe-Than and James D Herbsleb. 2019. Understanding hackathons for science: Collaboration, affordances, and outcomes. In *Information in Contemporary Society: 14th International Conference, iConference 2019, Washington, DC, USA, March 31–April 3, 2019, Proceedings 14*. Springer, 27–37.
[40] Joe Pitt-Francis et al. 2008. Chaste: using agile programming techniques to develop computational biology software. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366, 1878 (2008), 3111–3136.
[41] Adrian M Price-Whelan et al. 2022. The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5. 0) of the Core Package. *The Astrophysical Journal* 935, 2 (2022), 167.
[42] Naveen Raman et al. 2020. Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 57–60.
[43] Johannes Schindelin et al. 2015. The ImageJ ecosystem: An open platform for biomedical image analysis. *Molecular reproduction and development* 82, 7-8 (2015), 518–529.
[44] Judith Segal. 2008. Scientists and software engineers: A tale of two cultures. (2008).
[45] Rishab Sharma et al. 2022. Self-admitted technical debt in R: detection and causes. *Automated Software Engineering* 29, 2 (2022), 53.
[46] Naomichi Shimada et al. 2022. GitHub sponsors: exploring a new way to contribute to open source. In *Proceedings of the 44th International Conference on Software Engineering*. 1058–1069.
[47] Jefferson Silva et al. 2020. A theory of the engagement in open source projects via summer of code programs. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 421–431.
[48] Vandana Singh. 2012. Newcomer integration and learning in technical support communities for open source software. In *Proceedings of the 2012 ACM International Conference on Supporting Group Work*. 65–74.
[49] Igor Steinmacher et al. 2015. Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*. 1379–1392.
[50] Victoria Stodden and Sheila Miguez. 2013. Best practices for computational science: Software infrastructure and environments for reproducible and extensible research. *Journal of Open Research Software* (2013).
[51] Xin Tan et al. 2020. A first look at good first issues on GitHub. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 398–409.
[52] Xin Tan et al. 2022. An exploratory study of deep learning supply chain. In *Proc. Int'l Conf. Software Engineering (ICSE)*. 86–98.
[53] Erik H Trainer et al. 2014. Community code engagements: summer of code & hackathons for community building in scientific software. In *Proceedings of the 18th International Conference on Supporting Group Work*. 111–121.
[54] David Gray Widder et al. 2019. Barriers to Reproducible Scientific Programming. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Memphis, TN, USA, 217–221.