Understanding the Impact of Research Software Engineers on Scientific Software Development: A Mixed-Methods Study

Thomas Bock
Carnegie Mellon University
Pittsburgh, PA, USA

Will Sutherland
University of Washington
Seattle, WA, USA

Curtis Atkisson
University of Washington
Seattle, WA, USA

Anissa Tanweer
University of Washington
Seattle, WA, USA

James D. Herbsleb
Carnegie Mellon University
Pittsburgh, PA, USA

Bogdan Vasilescu
Carnegie Mellon University
Pittsburgh, PA, USA

Abstract

Research Software Engineers (RSEs) have emerged as a critical professional role bridging software engineering and scientific research. Despite rapid growth in RSE positions globally, their impact on scientific software quality and productivity remains understudied. We are conducting a mixed-methods study combining qualitative interviews with quantitative analysis of software repositories to understand how RSEs influence scientific software development. Our study examines RSE processes at research universities to understand collaboration challenges, impacts on software quality and sustainability, adoption of best practices, and effects on scientific productivity and knowledge transfer between RSEs and scientists. We present preliminary findings from this study, including both qualitative and quantitative data.

CCS Concepts

• **Software and its engineering → Collaboration in software development**; **Software creation and management**.

Keywords

Research Software Engineers, Scientific Software, Mixed Methods

ACM Reference Format:
Thomas Bock, Will Sutherland, Curtis Atkisson, Anissa Tanweer, James D. Herbsleb, and Bogdan Vasilescu. 2026. Understanding the Impact of Research Software Engineers on Scientific Software Development: A Mixed-Methods Study. In *1st International Workshop on Software Engineering and Research Software (SERS '26), April 12–18, 2026, Rio de Janeiro, Brazil*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3786172.3788360

1 Introduction

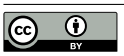
Science increasingly depends on software, with the quality of scientific software profoundly influencing the replicability, validity, and precision of scientific results [6]. The Research Software Engineer (RSE) role has emerged to address the growing complexity of scientific software development [1, 2]. RSEs bring deep software-engineering expertise to scientific projects, bridging the gap between traditional software development and scientific practice [12, 13]. While thousands of RSEs now work on scientific projects globally [5] and the rapid expansion of RSE recruitment strongly suggests that scientists and funding agencies see value in this approach, the extent of impact on technology development and scientific productivity has not been systematically investigated [4].

To fill this gap, we will address four key research questions in our full study: (1) What challenges do RSEs and scientists face in effective collaboration? (2) How does RSE collaboration impact software development processes and culture? (3) What is the technical impact on code quality and productivity? (4) What is the scientific impact on scholarly output and software reuse? Answering these questions is critical as funding agencies and research institutions increasingly invest in RSE positions and centers. Here, we present preliminary findings for this larger study. Specifically, we draw on preliminary qualitative and quantitative data to outline some dynamics and dimensions of variation between RSE groups that are relevant to the questions of collaboration and impact stated above.

2 Background

Challenges in Scientific Software Development. Scientific software development faces unique challenges that distinguish it from traditional software engineering. Requirements are often emergent rather than well-defined upfront, evolving as scientific understanding develops [12, 14]. Establishing shared understanding of software requirements across scientific and software-engineering domains and cultures proves difficult [13]. Scientists typically lack formal training in software-engineering practices, while software engineers may struggle to understand domain-specific scientific concepts and methods. The scientific "reputation economy" creates misaligned incentives, driving researchers to prioritize publications over the relatively low-payoff work of ensuring software sustainability beyond a particular publication [9]. Making scientific software publicly available presents additional challenges, as software intended for reuse must address generalized use cases beyond a particular publication, while also requiring considerable extra work such as documentation, tutorials, user support, and continued maintenance [10, 11, 15].

The RSE Movement. Establishing the RSE role began over a decade ago [1] and has experienced dramatic growth in recent years [5]. Professional associations like the US Research Software Engineer Association[1], the Society of Research Software Engineering[2] in the



SERS '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2404-6/2026/04
https://doi.org/10.1145/3786172.3788360

[1] https://us-rse.org/
[2] https://society-rse.org/

UK, or the German Society for Research Software[3] now provide advocacy, support, and resources to the profession [3, 7]. These organizations have focused on creating professional identity for RSEs and establishing recognition for their role in scientific projects. However, despite growth and institutional support, the impact of RSEs on technical development and scientific productivity has not been systematically investigated using rigorous empirical methods. To improve the understanding and impact of RSEs, researchers across the globe started to establish Research Software Engineering Research as a dedicated new research area [7], and so our study also falls into this research area.

3 Methodology

Research Design. Our study employs a mixed-methods approach combining qualitative interviews with quantitative repository analysis. Our preliminary data collection focuses on three privately-funded RSE teams at large research universities in the US. At these universities, RSEs work on centralized teams and engage research labs in short (3–15 months) projects, rather than being embedded in research labs long term. For quantitative analysis, we will also include a broader sample of RSEs, as we will describe below.

Qualitative Component. We will conduct approximately 80 interviews (one per person) with RSEs and RSE managers as well as researchers who worked with RSEs in the past. These will begin with purposive sampling of RSEs in dedicated RSE groups, sampling for variety in scientific domains, RSE team size, and the RSE's model of engagement with researchers (short rapid engagements versus being embedded long-term with a research group, for instance). We will then pursue snowball sampling of researchers who have worked with these groups in the past. Interviews will be semi-structured, with common questions across all interviews to allow for comparison as well as some questions tailored to different positions, such as for managers or for RSEs involved in short versus long-term engagements. Interview protocols and qualitative coding analyses will focus on challenges in scientist–RSE collaboration, domain-knowledge transfer, how RSEs connect scientists to software-engineering communities of practice, and the diffusion and persistence of software-engineering best practices. The qualitative component will also direct us towards aspects of the digital signature of RSEs (i.e., characteristics and behavioral aspects that can help us to identify RSEs, such as their tasks or working times) that we would otherwise miss.

Quantitative Component. We examine repository data from open-source scientific software projects, including commit histories, contributor information, issue discussions, and source-code evolution over time. Because RSEs are not always explicitly identified as such and sometimes work under different job titles (e.g., software developer, research programmer, etc.), it is not straightforward to identify people who take the role of an RSE in open-source software repositories and obtain a comprehensive view of their work [2, 8]. To identify RSEs and RSE-like contributors in these repositories, we will develop and validate heuristics, using ground-truth data from the privately-funded RSE teams and other sources. Using analytical techniques such as interrupted time-series analysis, we will measure the impact of RSEs on multiple dimensions: software practices (testing, continuous integration, code review, documentation), development activity (commit patterns, issue resolution speed), scientific outputs (publications mentioning the software, citation impact), and community adoption (software reuse, external contributions). To disambiguate common changes in software-development practices over time from those changes that occur specifically because of RSE involvement, we aim to compile matched control groups of scientific software projects without RSE involvement to support causal inference about RSE impact.

[3] https://de-rse.org/

4 Preliminary Results

At the time of submission, our study had just started. Our very preliminary data so far suggest that different institutions adopt distinct collaboration models. For example, team sizes vary significantly, from three RSEs supporting individual scientists to larger teams working on multi-institutional projects. Dependent on project scope and purpose, RSEs' working-time allocations also vary: While some RSEs adopt flexible working hours to accommodate scientists' schedules, in the majority of our 26 investigated projects, their commit patterns show that they tend to adhere to traditional 9-to-5 working hour schedules (in contrast to the majority of the scientists who contribute to the source code in these projects). At this time, these findings are specific to the teams investigated in this study and may not capture patterns of RSE contributions in other teams and contexts.

Our initial set of interviews (N=5) with organizers and managers of these RSE teams has also surfaced dynamics relevant to our larger questions of collaboration and impact. Firstly, our interlocutors described the demand on RSEs to perform multiple roles (to wear "different hats") as an important competency for making RSE projects go well. In particular, RSEs must manage frequent and ongoing user engagement and requirements development in addition to software-development work. Secondly, different RSEs had different levels of experience with both research processes and industry development processes, having differently understood benefits for enabling collaborations with researchers.

5 Expected Contributions and Outlook

With our research, we aim to make several important contributions to understanding the RSE role in scientific software development. Our work provides insights into the development processes, collaboration dynamics, and technical and scientific impacts of RSEs. Leveraging a mixed-methods approach, we triangulate findings from qualitative interviews and quantitative repository analysis to build a comprehensive picture of RSE influence. This will enable us to develop heuristics for identifying RSEs in open-source repositories, facilitating future large-scale studies. Our findings will inform researchers and funding agencies on the benefits and challenges of research software engineering in scientific ecosystems, targeting an improvement in the interaction between scientists and RSEs, fostering knowledge transfer, and promoting the adoption of software-engineering best practices in scientific research.

Acknowledgments

This work is supported by the Alfred P. Sloan Foundation and Schmidt Sciences (https://sloan.org/grant-detail/g-2025-25171).

References

[1] Rob Baxter, Neil Chue Hong, Dirk Gorissen, James Hetherington, and Ilian Todorov. 2012. The Research Software Engineer. In *Digital Research*.

[2] Alys Brett, Michael Croucher, Robert Haines, Simon Hettrick, James Hetherington, Mark Stillwell, and Claire Wyatt. 2017. Research Software Engineers: State of the Nation Report 2017. *Engineering & Physical Sciences Research Council (EPSRC)* (2017).

[3] Jeffrey C. Carver, Ian A. Cosden, Chris Hill, Sandra Gesing, and Daniel S. Katz. 2021. Sustaining Research Software via Research Software Engineers and Professional Associations. In *Int. Workshop on Body of Knowledge for Software Sustainability (BoKSS)*. IEEE, 23–24.

[4] Jeffrey C. Carver, Nic Weber, Karthik Ram, Sandra Gesing, and Daniel S. Katz. 2022. A Survey of the State of the Practice for Research Software in the United States. *PeerJ Computer Science* 8 (2022), e963.

[5] Ian A. Cosden, Kenton McHenry, and Daniel S. Katz. 2022. Research Software Engineers: Career Entry Points and Training Gaps. *Computing in Science & Engineering* 24, 6 (2022), 14–21.

[6] James H. Davenport, John Grant, and Catherine M. Jones. 2020. Data Without Software Are Just Numbers. *Data Science Journal* 19 (2020), 3.

[7] Michael Felderer, Michael Goedicke, Lars Grunske, Wilhelm Hasselbring, Anna-Lena Lamprecht, and Bernhard Rumpe. 2025. Investigating Research Software Engineering: Toward RSE Research. *Communications of the ACM* 68, 2 (2025), 20–23.

[8] Florian Goth, Renato Alves, Matthias Braun, Leyla Jael Castro, Gerasimos Chourdakis, Simon Christ, Jeremy Cohen, Stephan Druskat, Fredo Erxleben, Jean-Noël Grad, Magnus Hagdorn, Toby Hodges, Guido Juckeland, Dominic Kempf, Anna-Lena Lamprecht, Jan Linxweiler, Frank Löffler, Michele Martone, Moritz Schwarzmeier, Heidi Seibold, Jan Philipp Thiele, Harald von Waldow, and Samantha Wittke. 2025. Foundational Competencies and Responsibilities of a Research Software Engineer: Current State and Suggestions for Future Directions. *F1000Research* 13 (2025), 1429.

[9] James Howison and James D. Herbsleb. 2013. Incentives and Integration in Scientific Software Production. In *Proc. Int. Conf. Computer Supported Cooperative Work (CSCW)*. ACM, 459–470.

[10] Xing Huang, Xianghua Ding, Charlotte P. Lee, Tun Lu, and Ning Gu. 2013. Meanings and Boundaries of Scientific Software Sharing. In *Proc. Int. Conf. Computer Supported Cooperative Work (CSCW)*. ACM, 423–434.

[11] Andrew B. Neang, Will Sutherland, David Ribes, and Charlotte P. Lee. 2023. Organizing Oceanographic Infrastructure: The Work of Making a Software Pipeline Repurposable. *Proceedings of the ACM on Human-Computer Interaction (HCI)* 7, CSCW1 (2023), 79.

[12] Judith Segal. 2005. When Software Engineers Met Research Scientists: A Case Study. *Empirical Software Engineering (EMSE)* 10, 4 (2005), 517–536.

[13] Judith Segal. 2009. Software Development Cultures and Cooperation Problems: A Field Study of the Early Stages of Development of Software for a Scientific Community. *CSCW* 18, 5 (2009), 581.

[14] William Sutherland-Keller. 2025. *Research Software Systems: Exploration and Infrastructure in Observational Cosmology*. Ph. D. Dissertation. University of Washington.

[15] Erik H. Trainer, Chalalai Chaihirunkarn, Arun Kalyanasundaram, and James D. Herbsleb. 2015. From Personal Tool to Community Resource: What's the Extra Work and Who Will Do It?. In *Proc. Int. Conf. Computer Supported Cooperative Work (CSCW)*. ACM, 417–430.