Report from Dagstuhl Seminar 24161

Research Software Engineering: Bridging Knowledge Gaps

Stephan Druskat*[1], Lars Grunske*[2], Caroline Jay*[3], and Daniel S. Katz*[4]

1 **German Aerospace Center (DLR), Berlin, DE.** stephan.druskat@dlr.de
2 **HU Berlin, DE.** grunske@informatik.hu-berlin.de
3 **University of Manchester, GB.** caroline.jay@manchester.ac.uk
4 **University of Illinois Urbana-Champaign, US.** d.katz@ieee.org

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar "Research Software Engineering: Bridging Knowledge Gaps" (24161). The seminar brought together participants from the research software engineering and software engineering research communities, as well as experts in research software education and community building to identify knowledge gaps between the two communities, and start collaborations to overcome these gaps. Over the course of five days, participants engaged in learning about each others' work and collaborated in breakout groups on specific topics at the intersection between the two communities. Outputs from the working groups will be collected in a journal special issue and distributed via a dedicated website.

Seminar April 14–19, 2024 – https://www.dagstuhl.de/24161
2012 ACM Subject Classification Applied computing → Computers in other domains; Applied computing → Education; Software and its engineering
Keywords and phrases community building, Dagstuhl Seminar, knowledge transfer, research software engineering, RSE, software engineering research
Digital Object Identifier 10.4230/DagRep.14.4.42

1 Executive Summary

Stephan Druskat (German Aerospace Center (DLR), Berlin, DE)
Lars Grunske (HU Berlin, DE)
Caroline Jay (University of Manchester, GB)
Daniel S. Katz (University of Illinois Urbana-Champaign, US)

Research Software Engineering (*RSEng*) is the practice of applying knowledge, methods and tools from software engineering in research. Software Engineering Research (*SER*) develops methods to support software engineering work in different domains. The practitioners of research software engineering working in academia – Research Software Engineers (*RSEs*) – are often not trained software engineers. Nevertheless, RSEs are the software experts in academic research. They translate research to software, enable new and improved research, and create software as an important output of research [1].

* Editor / Organizer

Research Software Engineering: Bridging Knowledge Gaps, *Dagstuhl Reports*, Vol. 14, Issue 4, pp. 42–53
Editors: Stephan Druskat, Lars Grunske, Caroline Jay, and Daniel S. Katz
Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Hypothetically, the RSEng community and the SER community could benefit from each other. RSEs could leverage software engineering research knowledge to adopt state-of-the-art methods and tools, thereby improving RSEng practice towards better research software. Vice versa, software engineering research could adopt RSEng more comprehensively as a research object, to investigate the methods and tools required for the application of state-of-the-art software engineering in research contexts [2].

There are currently both unknown and known unknowns that make it hard for either community to attain the benefits mentioned above. We call these unknowns *gaps*, and we call methods to discover the unknown unknowns and to clarify or resolve the (subsequently) known unknowns *bridges*.

To find the gaps between research software engineering and software engineering research, and start building bridges between the two communities with the aim to create mutual benefit through reciprocal collaboration, we organized and held a five-day seminar in April 2024 at Schloss Dagstuhl – Leibniz Center for Informatics, as Dagstuhl Seminar 24161 "Research Software Engineering: Bridging Knowledge Gaps". Here, we report and document the seminar's program, outputs, and potential outcomes.

Seminar participants

In the past, there has been little focused direct communication between the RSEng and SER communities. Anecdotally, while the German RSEng and SER conferences co-located in 2023 in Paderborn and shared a break room, there was little, and only informal, exchange in session attendance by participants of either conference, and hence little knowledge exchange.

We specifically organized our Dagstuhl Seminar to serve as a bridge across this type of **communication gap** between the communities, by inviting international experts from both communities as well as individuals who have a track record of working at the intersection between the communities. We also invited experts in adjacent fields such as education and training, and research software funding.

In a pre-seminar survey, we asked participants if they could contribute experience in any of the following areas:

- *Research Software Engineering*: practicing software engineering in a research context
- *Software Engineering*: practicing software engineering
- *Research on Software Engineering*: conducting SER
- *Research on Research Software Engineering*: conducting research *on* RSEng
- *Teaching, training, instruction*
- *Research software funding*
- *Community building*

Answers were provided on a 5-point Likert scale: "None" – "A little" – "Some" – "Much" – "A lot". The results are shown in Figure 1. They suggest that we were mostly successful in bringing together participants with at least some relevant expertise, with the possible exception of research software funding, where more than half of participants claimed little or no expertise. This was partly due to invitees with a known background in research software funding being unavailable to attend the seminar.

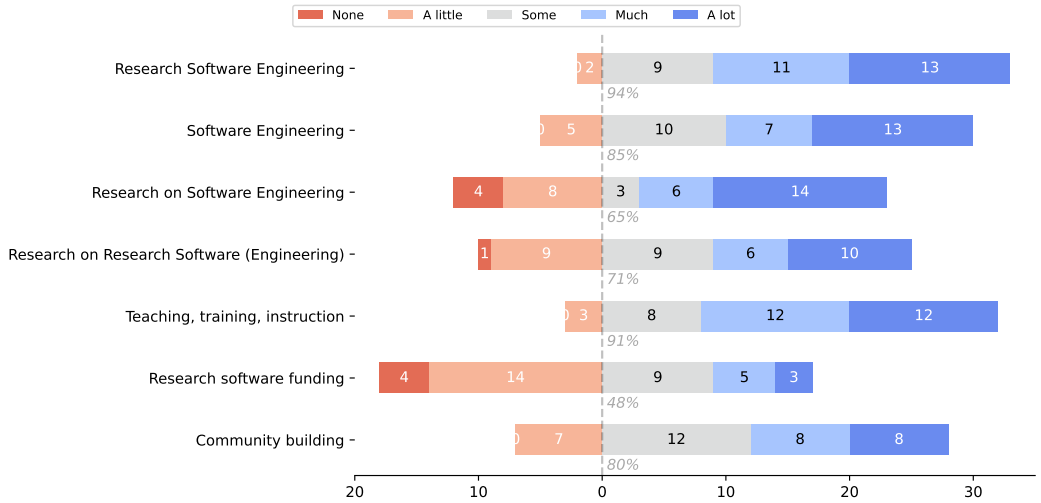

Figure 1 Survey responses to a question on experience in a given area. A total of 35 participants replied to the survey. Each person answered one of five levels of experience in the respective area.

Seminar program

The seminar program was prepared to mainly enable direct collaboration of participants from both the SER and the RSEng communities. Sessions were run either in the plenary, or in breakout groups.

After an introductory session where participants introduced themselves and their work, short presentations were given on ideas for collaborative groups to form at the seminar. From the original set of ideas, participants eventually formed a total of seven breakout groups to work on specific topics. Their work is summarized in the respective working group abstracts in this report. In addition, three additional ideas were brought forward in the course of the seminar in a more informal manner outside of sessions. For these additional ideas, work would either be done in parallel to the sessions, or was planned to be done after the seminar. An overview of all topics is given in Table 1.

Table 1 Topics of working groups and breakout initiatives worked on at the seminar.

Collaboration	Topic
Working group	Research Software Engineering Training & Education
Working group	Better Architecture, Better Software, Better Research
Working group	Bridging Communities
Working group	Developing a Common Language
Working group	Research Software: Towards Categories and Lifecycles
Working group	Security and usability of research software
Working group	Demystifying research software engineering for research group leaders
Breakout discussion	Software Engineering Equity, Diversity, Inclusion or Accessibility Research in Research Software/RSEng
Breakout project	Short software engineering social videos

The introductory session was rounded off by a primer presentation of research software engineering, and a preliminary discussion of options for the dissemination of seminar outcomes. Based on early feedback from participants, we decided to give up on our original idea of collecting seminar outcomes as chapters of an edited volume – a field manual for research software engineering. Instead, we discussed options for collecting some of the outputs in a journal special issue, while leaving it generally up to groups to determine the best way for their outputs to be disseminated.

For the remainder of the seminar, work was originally planned to take place in breakout groups, with regular reporting and feedback sessions in the plenary. The third day was planned as an exception, with two fishbowl sessions scheduled for the morning sessions, and a group excursion in the afternoon.

Based on feedback from participants on the first day of the seminar, we realized that there was a wider **perception gap** between RSEs and software engineering researchers than originally anticipated: software engineering researchers specifically did not feel entirely confident in their understanding of the scope and practice of RSEng, and of what research software projects looked like. Vice versa, RSEs did not feel entirely confident in their understanding of the aims and scope of software engineering research. We aimed to address this gap on the second day of the seminar with an ad-hoc "Ask us anything" plenary session where software engineering researchers asked members of the RSEng community questions about their work and experience, and with a subsequent ad-hoc session where RSEs presented the contexts they work in, setup of RSE groups, and particular research software projects.

In light of the need to address this kind of feedback quickly, we moved to adapt the original schedule for each day the night before, incorporating feedback we have had during the day.

In the same spirit as the "Ask us anything" sessions, we ran two sessions of "mythbusting fishbowls," where members of the SER, and then the RSEng, community formed a dynamic panel who discussed topics suggested by the respective other community: preconceptions that one community had about the other's work were fielded via `sli.do`, voted for by the audience, and the most popular ones then discussed by the panel. Panel members were replaced whenever another community member wanted to contribute to the discussion.

Outputs of the seminar activities are presented in brief in the next section.

Outputs

A central *outcome* of the seminar is that community members from the SER and RSEng communities started collaborations to identify and bridge gaps between them.

The outputs of the working groups will be invited to be submitted as articles to a special issue "Research Software Engineering: Discovering and Bridging Knowledge Gaps" in *IEEE Computing in Science & Engineering*, to be edited by the seminar organizers and published in 2025.

Beyond collecting outputs in this way, one of the working groups developed and published a website that aims to collect outputs from the seminar in a way accessible to a wider public and invites contributions from the community: `ser-rse-bridge.github.io`. The website also includes a mapping of terms between SER and RSEng, based on the Guide to the Software Engineering Body of Knowledge [3], which is currently under development (see Section 3.1).

Additionally, a series of short videos was produced during the seminar. In these videos, participants introduce central SER and RSEng knowledge concepts in under a minute. These can be used on social media platforms to create interest in these topics with, e.g., students looking to choose their courses.

Conclusion and future work

In conclusion, Dagstuhl Seminar "Research Software Engineering: Bridging Knowledge Gaps" (24161) was successful in bringing together members of two communities that have a vested interest in research software engineering: research software practitioners and software engineering researchers. Together with software education and community experts, we learned about each others' work and started conversations and collaborations.

Creating conversations between separate communities and their cultures and codes proved to be challenging at times, e.g., where incentives differed. Where we observed antagonism, or where it was brought to our attention by participants, we tried to defuse it and steer conversations into a constructive direction. We are confident that by and large, this worked well due to flexibility on the side of the participants and a general will to collaborate and progress.

We found that adapting the program to the needs of participants where possible while maintaining the general direction, and arguably intensive workload, helped make the seminar very productive and engaging. Participants have continued to engage after the seminar to identify gaps and potential bridges between the communities.

Future work should focus on the continuation of the efforts started at the seminar, and continued communication and collaboration between the communities. We believe that the seminar marked a starting point for collaboration that can realize future reciprocal benefit for research software engineering and software engineering research in equal measure. Interested parties can refer to the seminar website at https://ser-rse-bridge.github.io/ for related resources and activities.

References

1 J. Cohen, D. S. Katz, M. Barker, N. Chue Hong, R. Haines, and C. Jay, "The Four Pillars of Research Software Engineering," IEEE Software, vol. 38, no. 1, pp. 97–105, Jan. 2021, doi: 10.1109/MS.2020.2973362.

2 M. Felderer, M. Goedicke, L. Grunske, W. Hasselbring, A.-L. Lamprecht, and B. Rumpe, "Toward Research Software Engineering Research," Zenodo, Jun. 2023. doi: 10.5281/zenodo.8020525.

3 P. Bourque and R. E. Fairley, Eds., Guide to the Software Engineering Body of Knowledge, Version 3.0. IEEE, 2014.

2 Table of Contents

3 Working groups

3.1 Developing a Common Language: Mapping Between Software Engineering Fundamentals and Research Software Terminology

David E. Bernholdt (Oak Ridge National Laboratory, US), Robert Haines (University of Manchester, GB), Guido Juckeland (Helmholtz-Zentrum Dresden-Rossendorf, DE), Timo Kehrer (Universität Bern, CH), and Shurui Zhou (University of Toronto, CA)

URL https://ser-rse-bridge.github.io/mapping-of-terms/

When trying to build bridges between communities, it is critical that they are able to understand each other – that they "speak the same language". Often in science and technology, different communities tend to communicate more with each other than outside of the community. Over time, this leads a community to develop terminology that is distinctive. When people grounded in different communities meet, they may find that they don't speak the same language – the same word or term may mean different things in each community, and the participants in the conversation may not even realize it.

The term "software engineering" dates back to 1965 and software engineering research (SER) is a well-established academic discipline. Research Software Engineering (RSE), on the other hand, is a relatively young field (the term was coined in 2012) that, in this context, is primarily about applying the concepts, tools, and practices of software engineering to the development of research software (RS). The majority of research software engineers have come from the research software community, and despite the name, rarely have formal training in software engineering – rather than learned on demand as they've pursued their careers, often learning from others in their own community rather than seeking out resources produced directly by the SER community (classes, trainings, papers, etc.). As such the awareness and adoption of what the SER community would recognize as core concepts, practices, and tools, is variable, and often filtered through the experience of others in the RSE community – the two communities don't necessarily speak the same language.

To help build bridges between the two communities, we are developing a map between the terminologies of the SER and RSE communities, along with a rough assessment of the extent to which the RSE community is aware of the concept, the extent to which it is actually used, and the potential for research by the SER community to improve the use of the concept. We are using the Software Engineering Body of Knowledge (SWEBOK) as the jumping-off point for the SE fundamentals that we want to map. SWEBOK represents a systematic distillation of the field of software engineering by that community, though we expect that there will be additional terms arising in both communities that will need to be included in the mapping. The primary output of this work will be a living document, presented via a website, though we plan to summarize the results in papers, presentations, and other venues. We plan to engage first the participants of the Dagstuhl Seminar, and then reach out further in both the RSE and SER communities to flesh out the mapping. We will use the GitHub platform to carry out the discussion and track the revisions as we develop the map.

3.2 Security and usability of research software

Jeffrey Carver (University of Alabama, US), Stuart Allen (Cardiff University, GB), Hannah Cohoon (University of Utah – Salt Lake City, US), Anna-Lena Lamprecht (Universität Potsdam, DE), Christopher Klaus Lazik (HU Berlin, DE), Michael Meinel (DLR – Berlin, DE), and Lata Nautiyal (University of Bristol, GB)

This working group addressed the importance of security and usability in Research Software Engineering (RSE). The benefits of prioritising these quality attributes become apparent relatively late in the software development lifecycle, typically when developers are looking to expand their user base. However, this may be too late for efficient and effective implementation. There is clearly value in ensuring that software is designed from the outset to be both secure and user-friendly. Our work aims to improve awareness and skills related to security and usability in research software development. We find that the appropriate research methodologies for investigating security and usability in the context of RSE are quite similar. At the Dagstuhl Seminar, we conducted a pilot study to understand RSE/SER perspectives on these issues and to assess the level of awareness. These initial results show that despite the critical nature of usability, research software is often perceived as not usable and research software tools are often abandoned due to lack of usability. Furthermore, security is not necessarily perceived by seminar participants as an important quality of research software. We have begun a systematic review of the literature on security and quality in RSE and are planning further work to build on this initial effort, in particular conducting a larger survey and gathering testimonials through interviews.

3.3 Research Software: Towards Categories and Lifecycles

Mikaela Cashman McDevitt (Lawrence Berkeley National Laboratory, US), Michael Felderer (DLR – Köln, DE), Michael Goedicke (Universität Duisburg – Essen, DE), Wilhelm Hasselbring (Universität Kiel, DE), Daniel S. Katz (University of Illinois Urbana-Champaign, US), Frank Löffler (Friedrich-Schiller-Universität Jena, DE), Sebastian Müller (HU Berlin, DE), and Yo Yehudi (Open Life Science – London, GB)

URL https://github.com/ser-rse-bridge/RSE-lifecycle

There is a huge variety of types of research software, at different stages of evolution. This often confuses potential software users, developers, funders, and other stakeholders who need to understand a particular software project, such as when deciding to use them, contribute to them, or fund them. We present work performed by a group who met at a Dagstuhl Seminar consisting of both software engineering researchers (SERs) and research software engineers (RSEs). It includes an initial categorization of research software types, and an initial presentation of an abstract research software lifecycle that can be applied and customized to suit a wide variety of research software types, which then can be used to make decisions and guide development standards that may vary per stage. We also seek community input on improvements of these two artifacts for future iterations.

In addition, because terminologies and definitions often vary, e.g., one person may consider a software project to be early-stage or in "maintenance mode", whilst another project might consider the same software to be inactive or failed. Because of this, we explore and explain concepts such as software maturity, intended audience, and intended future use.

3.4 Better Architecture, Better Software, Better Research

Myra B. Cohen (Iowa State University – Ames, US), Neil Chue Hong (University of Edinburgh, GB), Stephan Druskat (German Aerospace Center (DLR), Berlin, DE), Nasir Eisty (Boise State University, US), Michael Felderer (DLR – Köln, DE), Samuel Grayson (University of Illinois – Urbana-Champaign, US), Lars Grunske (HU Berlin, DE), Wilhelm Hasselbring (Universität Kiel, DE), Jan Linxweiler (TU Braunschweig, DE), and Colin Venters (University of Huddersfield, GB)

In this breakout, we discussed the notion that better architecture leads to better research. Research software engineering requires flexible and modular architectures to accommodate rapid evolution and interdisciplinary collaboration. Hence, we argue that research software engineering should focus on architectural metrics to evaluate and improve their code. Architectural metrics in research software are essential for ensuring the software's scalability, performance, maintainability, and overall quality, facilitating reproducible and reliable research outcomes. We already have many metrics and tools to measure and improve the quality of software architecture. However, we hypothesized that research software is often built using limited resources, and without a long-term vision for maintainability, which may lead to architectural decay. In this breakout, the group (consisting of software engineers, software engineering researchers, and research software engineers) discussed key architectural metrics such as code smells, duplication, test coverage, cyclometric complexity, etc., and how these can be used to improve research software. We explored our hypothesis by applying existing architectural analysis tools to a few open-source research software repositories during our breakouts. We discovered high cyclomatic complexity, large god classes that need refactoring, and low test coverage. We concluded that we should explore this idea further and that there may be an opportunity to build better tools and techniques to help research software engineers improve the architecture of their software, which in turn can improve its quality.

3.5 Bridging Communities: Bringing the Research Software Engineering and Software Engineering Researcher Communities Together for Mutual Benefit

Ian Cosden (Princeton University, US), Jeffrey Carver (University of Alabama, US), Hannah Cohoon (University of Utah – Salt Lake City, US), Stephan Druskat (German Aerospace Center (DLR), Berlin, DE), Nasir Eisty (Boise State University, US), Carole Goble (University of Manchester, GB), Samuel Grayson (University of Illinois – Urbana-Champaign, US), and Samantha Wittke (CSC Ltd. – Espoo, FI)

As the other work from this Dagstuhl Seminar illustrates, there is a chasm between the community of Software Engineering Researchers (SERs) who cater mostly to industry applications and Research Software Engineers who may or may not have formal training in software engineering but develop code for research applications. However, we have identified a number of potential opportunities if we can bridge that chasm: SERs may find novel research questions from RSE experiences, and RSEs could improve their productivity by applying approaches and tools developed by SERs.

Change does not happen on its own. Rather, it must be encouraged and catalyzed by an initial vanguard group and eventually the whole community or communities involved. Therefore, it is incumbent upon those desiring such change to apply concepts from the theory of change, push motivational incentive levers love (open development), power (influence), money (funding), fame (recognition) and create the facilitating conditions of building trust, providing thrust through resources and support, and operating with transparency).

One community observing the other from a distance is a start but not sufficient for lasting change because it treats others as a means to a selfish end. Once community addressing the other is better, but still not enough because there is no feedback from the other to the one. Only true collaboration with cyclic communication, mutual benefit, and shared experiences will be enough for lasting change.

Recognizing the SER and RSE communities have developed and evolved independently, creating bridges between the two represents a cross-disciplinary and cross-cultural endeavor just as significant as between life science and computer science [1]. With this recognition we are developing a guide, as a separate publication, for parties from both sides to better understand how to foster new collaborations between the two communities. This guide, "10 Simple Rules for catalyzing collaborations and building bridges between RSEs and SERs" will outline some of the potential benefits while giving a simple set of rules to follow for both communities to thrive in a new, mutually beneficial collaboration.

References

1 Knapp B, Bardenet R, Bernabeu MO, Bordas R, Bruna M, Calderhead B, et al. (2015) "Ten Simple Rules for a Successful Cross-Disciplinary Collaboration". PLoS Comput Biol 11(4): e1004214. https://doi.org/10.1371/journal.pcbi.1004214

3.6 Demystifying research software engineering for research group leaders

Toby Hodges (The Carpentries – Oakland, US), Stuart Allen (Cardiff University, GB), Neil Chue Hong (University of Edinburgh, GB), Stephan Druskat (German Aerospace Center (DLR), Berlin, DE), Lars Grunske (HU Berlin, DE), Daniel S. Katz (University of Illinois Urbana-Champaign, US), Jan Linxweiler (TU Braunschweig, DE), Frank Löffler (Friedrich-Schiller-Universität Jena, DE), Jan Philipp Thiele (Weierstraß Institut – Berlin, DE), and Samantha Wittke (CSC Ltd. – Espoo, FI)

Unfamiliarity among principal investigators with some of the most important principles of research software engineering remains one obstacle to successful integration of software engineering practices into research. Research group leaders unfamiliar with essential concepts and practices in (research) software engineering may find it difficult to provide guidance to RSEs in their projects/groups, or to leverage their expertise effectively as part of the research process. While a growing body of literature, training materials, and other resources exists to help novice research software engineers learn key principles and develop good practices, one reason for the enduring knowledge gap among research group leaders may be that they are not the target audience of such literature, lacking first-hand experience of computational research methods. Often, these resources do not frame RSEng skills within the context of the research process as a whole. Inspired by the popular "Ten Simple Rules" series of articles in PLOS CompBio, this group aims to inform PIs and other researchers about good practices in RSEng – e.g. software testing, documentation, version control, modelling software architecture – and explain the value of these when applied by an RSE to enrich research.

Participants

- Stuart Allen
Cardiff University, GB
- David E. Bernholdt
Oak Ridge National Laboratory, US
- Jeffrey Carver
University of Alabama, US
- Mikaela Cashman McDevitt
Lawrence Berkeley National Laboratory, US
- Neil Chue Hong
University of Edinburgh, GB
- Myra B. Cohen
Iowa State University – Ames, US
- Hannah Cohoon
University of Utah – Salt Lake City, US
- Ian Cosden
Princeton University, US
- Stephan Druskat
German Aerospace Center (DLR), Berlin, DE
- Nasir Eisty
Boise State University, US
- Michael Felderer
DLR – Köln, DE
- Carole Goble
University of Manchester, GB
- Michael Goedicke
Universität Duisburg – Essen, DE
- Samuel Grayson
University of Illinois – Urbana-Champaign, US
- Lars Grunske
HU Berlin, DE
- Robert Haines
University of Manchester, GB
- Wilhelm Hasselbring
Universität Kiel, DE
- Toby Hodges
The Carpentries – Oakland, US
- Caroline Jay
University of Manchester, GB
- Guido Juckeland
Helmholtz-Zentrum Dresden-Rossendorf, DE
- Daniel S. Katz
University of Illinois Urbana-Champaign, US
- Timo Kehrer
Universität Bern, CH
- Anna-Lena Lamprecht
Universität Potsdam, DE
- Christopher Klaus Lazik
HU Berlin, DE
- Jan Linxweiler
TU Braunschweig, DE
- Frank Löffler
Friedrich-Schiller-Universität Jena, DE
- Michael Meinel
DLR – Berlin, DE
- Sebastian Müller
HU Berlin, DE
- Lata Nautiyal
University of Bristol, GB
- Bernhard Rumpe
RWTH Aachen, DE
- Heidi Seibold
München, DE
- Jan Philipp Thiele
Weierstraß Institut – Berlin, DE
- Colin Venters
University of Huddersfield, GB
- Samantha Wittke
CSC Ltd. – Espoo, FI
- Yo Yehudi
Open Life Science – London, GB
- Shurui Zhou
University of Toronto, CA