Software engineering, research software and requirements engineering

Stephan Druskat[1], Michael Felderer[1,2], and Carina Haupt[1]

[1]German Aerospace Center (DLR), Institute for Software Technology, {12489 Berlin, 51147 Cologne}, Germany

[2]University of Cologne, Department of Mathematics and Computer Science, 50923 Cologne, Germany

{stephan.druskat,michael.felderer,carina.haupt}@dlr.de

Abstract

Research software is developed with aims and in contexts that differ from other software, by individuals who are often not trained software engineers. Furthermore, research software projects often suffer from resource scarcity, but produce artifacts that are crucial for research and require software engineering. These circumstances have an impact on how software engineering activities are performed in research contexts. This holds in particular for requirements engineering, which plays a crucial role for the success of traditional software engineering projects. In this paper, we discuss the relationship between research software and requirements engineering. For that purpose, we also reflect on research software engineering and its relationship to software engineering research. Closer collaboration between software engineering research and research software engineering creates opportunities for knowledge exchange and better research software engineering practice, and interesting new research questions for software engineering research. This is true also for requirements engineering, where lightweight methods are required, depending on the type of research software developed.

1 Introduction

Research in academia and industry to a great extent relies on software that enables or supports the research process. In many cases, this software is *research software*: software that is either created during the research process itself, or developed specifically for a research purpose [6]. The main aim of research software is creating and validating knowledge, which sets it apart from other types of software such as business software. As such, software today is a ubiquitous and essential element of academic and industrial research.

Originally, however, research software was not considered a relevant object of software engineering. The NATO Software Engineering Conference in Garmisch in 1968 focused on software engineering of mission-critical business and embedded software. Significantly, the conference report excludes "researchers in fields other than software engineering and computer science" from the report's primary audience as a group that "need[s] an understanding of the nature of software engineering, although they are not themselves working in the field." [9, p. 9].

Despite this initial exclusion of scientific data processing applications in computational science as a principal object of software engineering, a new discipline of *Research Software Engineering* has emerged.

This paper is structured as follows. Section 2 defines and discusss the area of research software engineering. Section 3 reflects on the relationship between research software engineering and software engineering research. Section 4 discusses the role of requirements engineering for research software. Finally, Section 5 concludes this paper.

2 Research Software Engineering

Broadly speaking, research software engineering[1] is the use of software engineering practices in research applications. It is performed by *Research Software Engineers (RSEs)* in many academic research disciplines, and to a lesser extent in industrial (research) settings.

The RSE role is diverse across a number of dimensions. While it generally covers traditional software engineering as defined in the Guide to the Software Engineering Body of Knowledge (SWEBOK), RSEs apply the relevant practices in research-specific fields such as modeling and simulation, data science, DevOps and ResearchOps, porting and optimisation, but also traditional application development. They do this across the whole spectrum of research disciplines and often in inter-disciplinary contexts. In terms of their working environment, RSEs can be part of a central RSE team at a research institution, embedded in a specific research group or project, or somewhere in between.

One of the main challenges in research software engineering is a lack of formal training in software engineering. According to the 2022 International RSE

[1]See also *GI Radar 351: Research Software.*

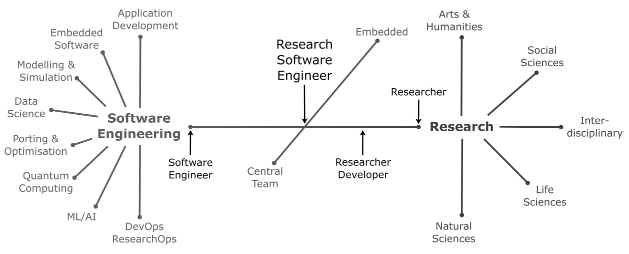

Figure 1: Dimensions of diversity in the Research Software Engineer (RSE) role (adapted from [2]).

Survey [7], the majority (~74%) of just over 1,000 self-identifying RSEs worldwide do not have a background in computer science. And they spend their time on a number of other tasks beside software development, see Figure 2. Approximately 42% of RSEs do not see themselves as professional developers. Additionally, many RSEs develop software for themselves rather than for others, with only ~26% developing software "mostly for other people". Around half of the surveyed population of RSEs also work outside of a dedicated RSE group, with a project bus factor of 1, indicating that their knowledge will be lost once they leave the project or institution.

There are, however, exceptions to this rule, notably at our own institution, the German Aerospace Center (DLR)[2]. At DLR, we have established a long-standing community for RSEs through the Software Engineering Network, with representatives at all 61 institutes and facilities. In addition, we provide guidelines for software development [10], tooling infrastructure, and trainings and consulting to research projects.

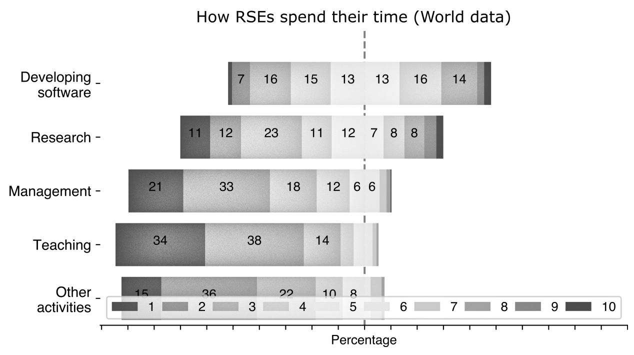

Figure 2: Time spent by RSEs across the world on different tasks (Likert scale from 0 (None) to 10 (All the time)), from [7].

Another issue in research software engineering, generally, is the unique context in which research software is developed. Research is often highly experimental, which means that functional requirements will often be unknown, or may change rapidly alongside an experimental research process. As research software aims to create knowledge in the first place, verification and validation are difficult as long as this knowledge is either non-existent or merely hypothetical, that is, until the end of a research cycle. Adopting structured software engineering processes to the development of research software may furthermore restrict the required agility of the research process.

[2]https://rse.dlr.de

The research environment, especially in academic research, also defines unique non-functional requirements. Research results must be reproducible, that is, it must be possible for other to validate the knowledge gained through computational research. This has a strong impact on how research software must be documented, but also how artifacts are treated. The FAIR Principles for Research Software [3] require that research software must be: **F**indable by humans and machines; **A**ccessible through standardised protocols; **I**nteroperable by way of data exchange or APIs; **R**eusable, i.e., it can be understood, modified, built upon or incorporated into other software. On top of this, research software may be long-lived, but is mainly developed by individuals with fixed-term contracts: PhD students, post-docs, etc.

Taken together, all of these factors introduce additional weights to the well-known trade-off between time, resources, functionality and quality in software engineering (Figure 3).

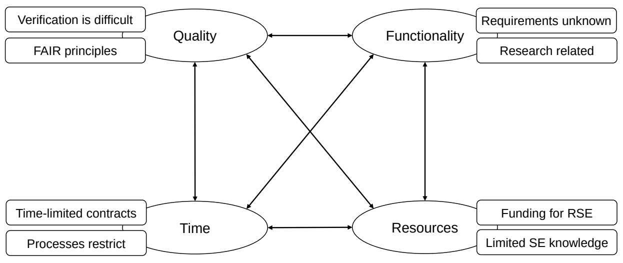

Figure 3: The additional weights in research software engineering in the trade-off between time, resources, functionality and quality in software engineering.

3 Research software engineering and software engineering research

With the rise of the RSE role in the UK, where the first International Conference of Research Software Engineers took place in 2016, Research Software Engineering is becoming increasingly organized. National or multinational RSE associations have been founded in the UK, the Netherlands, Germany, Belgium, the Nordics, Australia and New Zealand, the US, and in Denmark. Of these, the UK, German, Australian and New Zealand, and US associations run annual national RSE conferences. The associations play a pivotal role in establishing the RSE role in their respective contexts, building communities, organizing knowledge exchange and training, and building bridges to other communities, including the software engineering research community.

The RSE and software engineering research (SER) communities have a lot to gain from working together closely. On the one hand, research software is a critical artifact that requires software engineering. SER can provide state-of-the-art software engineering knowledge to the RSE community. Examples for this are automated metamorphic testing of software that has no clearly defined test oracles [8], or agile and continuous software development processes.

On the other hand, RSE can provide interesting new research questions to SER that reflect unique properties of research software, e.g., around the organization of software-centric research processes, required skills and educational formats for RSEs, quality assurance for and maturity levels of research software, suitable business models for open source research software, etc.

It is therefore important to note promising new initiatives forming at the intersection between both fields. One of these is the Gesellschaft für Informatik's SIG "Research Software Engineering"[3] as part of the section Software Engineering. This group has spawned several working groups for, e.g., making research software accessible for different activities through categorization, developing research programmes for "RSE Research" (see also [5]), or the creation of a guideline to implement research software engineering guidelines for research institutions.

Another effort is the Dagstuhl Seminar "Research Software Engineering: Bridging Knowledge Gaps"[4], that takes place from 14–19 April 2024 and brings together experts from RSE and SER to work on improving the knowledge exchange between the two fields and foster further collaboration. One of the main topics of this seminar will be the effective specification of the requirements of research software in different application and research disciplines.

And indeed, against the background established in this article so far – where functional requirements for research software may be unidentifiable, and non-functional requirements are imposed by the research context – it is interesting to look at the challenges and possibilities of requirements engineering for research software in more detail.

4 Research software and requirements engineering

In research software engineering, reproducibility of research results is key. This is relevant especially where decisions that affect many people are made based on the results of computational research. One recent example of this is the modeling and simulation of pandemic development during the COVID-19 pandemic.

As a tool for risk management, requirements engineering can help identify and mitigate risks, and reduce the likelihood of failures. Therefore it seems natural that in situations where research software is developed to create and validate the knowledge that underpins important decisions, requirements engineering plays an important role in the development of research software. While this is certainly true for traditionally safety-relevant research software projects, e.g., in applied aeronautics and space research and other engineering disciplines, or in medical research, state-of-the-art requirements engineering processes from software engineering research are not applied across the board in research software engineering. Anecdotal evidence informally elicited by software developers from mostly university-based research software projects points partially to a lack of knowledge about requirements engineering as a process, partially to a low priority of requirements engineering in the development process due to conflicting incentives, and partially to the ad-hoc implementation of "issue-based requirements tracking".

In other cases, requirements engineering is practiced informally or only in part. The research software project HERMES[5] develops a demonstrator for continuous integration software that enables the automatic publication of FAIR research software with rich metadata in long-term repository archives. The project organized several workshops over the project runtime. These included a kick-off meeting with key stakeholders, as well as interactive workshops with users where requirements elicitation and feedback loops were closely paired with development sprints. Some high-level requirements were also documented in a concept paper early on in the development process [4]. While requirements were frequently discussed in project meetings, they were not formally managed, analysed or traced.

There are multiple reasons why requirements engineering is not applied everywhere in research software engineering, of which some have already been touched upon above: research projects often lack the resources for professional research software engineering, and many RSEs are not trained software engineers, much less trained requirements engineers. Additionally, due to missing knowledge transfer between software engineering research and research software engineering, requirements engineering is often still understood as a documentation-heavy process that requires waterfall development processes. And on top of this, requirements engineering may generally not be considered an option, based on the assumption that especially functional requirements cannot be known due to the experimental nature of the research in which software is developed.

These latter assumptions disregard the fact that not all research software is the same. Previous and current classification efforts seem to suggest that, indeed, some types of research software may not lend themselves easily to some highly formalized requirements engineering processes. These may include the implementation of novel methods and models, or early prototypes that are used only by individual researchers. Others, however, are not dissimilar to

[3]https://fg-rse.gi.de
[4]https://dagstuhl.de/24161
[5]https://software-metadata.pub

software applications developed in business contexts. These may include proof-of-concept implementations, and accepted methods and models that are used by whole research communities or are adapted to new research needs. And certainly, instrument or other embedded software, as well as research platforms and infrastructure can be submitted to even the more formal established requirements engineering processes.

The above-mentioned assumptions also ignore the fact that software engineering research has developed and continues to develop more modern and lightweight requirements engineering processes that support agile paradigms much better than traditional processes. RSEs already often use collaborative development and DevOps platforms such as GitHub and GitLab, and would be well-placed to adopt requirements engineering paradigms that leverage such tooling. And vice versa, requirements engineering research will also be inherently interested in trying to tackle the specific challenges of research software engineering, and pursue solutions that also take into account the non-functional requirements of research and open science contexts.

Finally, there may be an opportunity to create synergies from combining the documentation requirements for software from requirements engineering, and the documentation requirements for research in general. In the end, research software projects – and research projects more generally – always work with stakeholders, first and foremost the researchers themselves, but also research funders, politics and the general public. They already document requirements in project proposals and track them in research publications. There are also potentials for synergies with respect to the type of software developed. The nature of research software (in particular with respect to research software that focuses on modeling, simulation and data analytics) has similarities and synergies with the development of machine-learning-based software components - and in the future maybe also with quantum-based software components - in industry [1]: requirements are typically not-known upfront and therefore testing heavily relies on randomization, exploration, runtime monitoring, and specific approaches like metamorphic testing [8]. Such approaches are also essential for research software, which therefore provides suitable evaluation contexts for requirements engineering and testing of (industrial) machine-learning-based software components. Therefore, requirements engineering and testing for research software has synergies with the respective activities for industrial machine-learning-based software systems.

5 Conclusion

Software is a critical artifact of research that requires research software engineering. Research software engineering and the RSE role, however, are highly diverse, and we currently do not know enough about either. This makes RSE research necessary and presents opportunities for new questions in software engineering research. Requirements engineering research in particular can offer knowledge transfer to fill existing gaps, and can develop solutions to the particular challenges of research software engineering. These could build on the tools already available to RSEs, and harness existing documentation processes in research.

Acknowledgments

SD acknowledges funding for HERMES (ZT-I-PF-3-006) by the Helmholtz Initiative and Networking Fund through the Helmholtz Metadata Collaboration.

References

[1] Jubril Gbolahan Adigun et al. "Collaborative Artificial Intelligence Needs Stronger Assurances Driven by Risks". In: *Computer* 55.3 (2022), pp. 52–63.

[2] Neil Chue Hong. *Is Research Software Engineering Coming of Age?* Presentation. Sept. 2023. DOI: 10.6084/m9.figshare.24078054.v5.

[3] Neil P. Chue Hong et al. *FAIR Principles for Research Software (FAIR4RS Principles).* Tech. rep. Research Data Alliance (RDA), May 2022. DOI: 10.15497/RDA00068.

[4] Stephan Druskat et al. "Software Publications with Rich Metadata: State of the Art, Automated Workflows and HERMES Concept". In: *arXiv* (2022). DOI: 10.48550/arXiv.2201.09015.

[5] Michael Felderer et al. *Toward Research Software Engineering Research.* Tech. rep. Zenodo, June 2023. DOI: 10.5281/zenodo.8020525.

[6] Morane Gruenpeter et al. *Defining Research Software: A Controversial Discussion.* Tech. rep. Zenodo, Sept. 2021. DOI: 10.5281/zenodo.5504016.

[7] Simon Hettrick et al. *International RSE Survey 2022.* Zenodo. Aug. 2022. DOI: 10.5281/zenodo.7015772.

[8] Upulee Kanewala, Anders Lundgren, and James M. Bieman. "Automated Metamorphic Testing of Scientific Software". In: *Software Engineering for Science.* Chapman and Hall/CRC, 2016.

[9] Peter Naur and Brian Randell. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO.* 1969.

[10] Tobias Schlauch, Michael Meinel, and Carina Haupt. *DLR Software Engineering Guidelines.* Tech. rep. Zenodo, 2018. DOI: 10.5281/ZENODO.1344612.