DOI:10.1145/3685265

Wilhelm Hasselbring et al.

Opinion

Investigating Research Software Engineering: Toward RSE Research

Research software engineering research aims at understanding and improving how software is developed for research.

RESEARCH SOFTWARE IS software designed and developed to support research activities. It can be used to collect, process, analyze, and visualize data, as well as to model complex phenomena and run sophisticated simulations. Research software is developed by researchers themselves or by software developers working closely with researchers. Research software is typically developed to meet specific research needs, and it often has unique requirements that are different from standard commercial software.

Research software engineering (RSE) and the related role of research software engineer has emerged as a job profile in its own right. We highlight the concept of research software engineering research—RSE research in short—as a complementary approach to RSE: conducting research on understanding and improving how software is developed for research. We start with a look at 50 years of software engineering research, before introducing the characteristics of research software, RSE in general, RSE research, and then concluding with an outlook to further essential activities on RSE research.

Software Engineering Research

Randell[14] reviewed the first 50 years of software engineering: To address the so-called software crisis, NATO was the sponsor of the first software engineering conference in 1968. The perception at that time was that while errors in scientific data processing applications might be a nuisance, they are to a large extent tolerable. In contrast, failures in mission-critical military systems might cost lives and substantial amounts of money. Based on this attitude, software engineering—like computer science as a whole—aimed for generality in its methods, techniques, and processes and focused almost exclusively on business and embedded software, and system software, such as operating systems, networks, and compilers. Software development is more than just programming. Meanwhile, the software engineering discipline has gained a lot of insight into the whole process of software development, accumulating in various textbooks on software engineering, a multitude of books on dedicated sub-fields

Research software engineering (RSE) is a discipline that focuses on the development of software for research purposes.

of software engineering, such as requirements engineering, software architecture, design, modeling, testing, and development processes. The Software Engineering Body of Knowledge (SWEBOK) structures and aggregates what software developers have learned in the last 50+ years.

Software engineering does not only have these sub-disciplines mentioned previously, which cover the different activities within a software engineering project. To properly address the various application areas, software engineering is also organized into domain-specific sub-disciplines, such as automotive software engineering. However, software engineering research largely ignored the specific demands of research software,[9] and vice versa.

Research Software Categories and Sustainability

Research software mainly falls into one of the following three top-level role categories (and sometimes combinations)[5]:

1. Modeling, Simulation, and Data Analytics of, for example, physical, chemical, social, linguistic, or biological processes in spatiotemporal contexts.

2. Technology Research Software in science and engineering research.

The goal of RSE is to support and advance research fields by providing high-quality software that is reliable, efficient, and easy to use.

3. Research Infrastructure Software, such as research data and software management systems.

The assignment of research software to categories may evolve over time. For instance, software specifically developed for a research question (usually Categories 1 and 2) can later turn into research infrastructure software (Category 3). In different contexts, software may also be in multiple categories at the same time.

Research software includes source-code files, algorithms, scripts, computational workflows, and executables created during the research process or for a research purpose. Software components (for example, operating systems, programming languages, libraries, and so forth) used for research but not created during research or with a clear research intent should be considered "software in research" and not "research software."[2] Research software should be FAIR[3,10] and open.[4] The goal of FAIR (Findable, Accessible, Interoperable, Reusable) is to increase the transparency, reproducibility, and reusability of research. Many research institutions are now recognizing the increasing importance of research software and are providing support for its development, sharing, and long-term preservation, but also development and maintenance effort. One of the main challenges with research software prototypes is the continuous maintenance and managed evolution[15] as open source by a research community, especially for that software, which was developed with some initial, limited funding. To address these challenges, the Software Sustainability Institute (SSI) was founded in 2010 in the U.K. as the first organization dedicated to improving software in several research domains.[1]

Research Software Engineering

Research software engineering is a discipline that focuses on the development of software for research purposes. It is a relatively new field that emerged in response to the growing recognition of the importance of research software and the need for specialized skills and expertise in its development. Overall, the goal of RSE is to support and advance research fields by providing high-quality software that is reliable, efficient, and easy to use. Specific to engineering research software is, for instance, that requirements are usually not known up front, emerge and evolve as research advances, and often hard to comprehend without some Ph.D. in science. Verification and validation are difficult and strictly scientific. Overly formal software processes restrict research. Few scientists are trained in software engineering, which leads to a disregard of most modern software engineering methods and tools. This situation created a chasm between software engineering and computational science.[7]

As early scientific software was developed by small teams of scientists primarily for their own research, modularity, maintainability, and team coordination could often be neglected without a large impact. As a consequence of the expected sustainability and reproducibility requirements on research software, the software turns from a by-product into a long-living, sustainable asset if not into a core research infrastructure, where demands for quality of the code heavily increase. This includes understandability, documentation, reuse, ability to evolve, adaptability, and other typical quality attributes that software engineering has discussed as a consequence of the software crisis in other development domains over the past 50+ years. However, software engineering approaches will only be adopted by scientists if these approaches honor the distinct characteristics and constraints of scientific software development.

Research software engineers develop, optimize, and maintain research software. They should have a deep understanding of both software engineering and research practices, and are skilled at bridging the gap between these two domains. To foster RSE skills and leverage the job profile of RSE, several associations have been founded, such as the U.K. Society of Research Software Engineering (https://society-rse.org/), the U.S. Research Software Engineer Association (https://us-rse.org/), or the German Society for Research Software (https://de-rse.org), which in turn coordinate in an international council (researchsoftware.org/council.html). The U.K. Software Sustainability Institute (https://www.software.ac.uk/) provides support for furthering RSE practice, while the Research Software Alliance (https://researchsoft.org) works toward evolving RSE policy by collaborating with decision makers and key stakeholders. We propose to complement these activities via RSE research.

Due to the exploratory nature of research, functional requirements are often unknown up front for research software.

Research Software Engineering Research

RSE research aims to improve the scholarly understanding of research software development, and to develop methods, techniques and tools to improve RSE practice. Therefore, RSE research must tackle a number of research questions, which do not only address the software as a product, but also software development assistance for the domain researcher as a human being with limited time and usually improvable skills for software development. Possible research questions include:[11]

- What is specific about RSE compared to other SE specializations?
- How to (better) organize software-centric scientific processes?
- How to (better) integrate SE techniques such as requirements engineering, architectural modeling and automated testing in the research software development process?
- What additional tools do RSE practitioners need?
- Which skills are required for RSE practitioners, and what are suitable education formats?
- What is the (potential) role of generative AI in RSE practice?

Let us take a look at an example RSE research area. Requirements engineering is crucial for software projects, but explicit requirements engineering is often ignored in scientific software projects. Due to the exploratory nature of research, functional requirements are often unknown up front for research software, which also makes verification and validation a challenging task. A research approach specifically designed for research software may extract requirements from available knowledge sources, such as related scientific papers, user manuals, and project reports.[12] An implication of unknown requirements is the test oracle problem for research software, which also requires specific solutions.[8] We acknowledge that Heroux[6] recently introduced a very similar notion of research software science. We fully consent with its goals, but feel that the term suggests research on research software as the study object, rather than research on the RSE process that leads to the software. To emphasize the research focus on these engineering aspects, we propose the concept of RSE research instead.

Conclusion and Outlook

Research software should be flexible and modular, allowing researchers to easily modify, compose, configure, and extend it as their research evolves. To ensure the quality and reproducibility of research, it is important that research software is well documented, tested, and maintained. Among the methods and techniques that software engineering can offer to RSE are model-driven software engineering with domain-specific languages, modular software architectures, specific requirements engineering techniques, and testing without test oracles.[7] Research software engineering correctly pushes these techniques into development projects for researchers, but domain-specific adaptations are necessary. This way, research may achieve maintainable, sustainable software, in particular for community research software. We suggest addressing this goal via empirical software engineering research.[13]

Software development tools for the automation of various activities and wizard-like assistance have enormously increased productivity and simplified the hurdles for newcomers to create software. Moreover, these tools nowadays enable non-experts to develop significant pieces of software and leverage the knowledge of core software techniques, such as persistent storage, communication, compilation, computation orchestration, and so forth.

Researchers must be aware that software engineering is not only about getting the code right but also involves architectural, design, quality assurance, and management soft skills to be adopted and lived during a development process. Researchers who create software for a sustainable, long-lasting infrastructure must be trained in software engineering skills, which drastically differ from mere programming skills.

We have learned that there are generic software engineering techniques that can be applied in many domains, but due to the domain-specific differences in characteristics, it is also useful to adapt, enhance and possibly create domain-specific techniques, tools, methods, frameworks, and so forth. It is necessary to build better domain-specific tooling to address the domain-specific challenges of research software and to establish RSE Research as a research field over RSE.

Researchers must be aware that software engineering is not only about getting the code right but also involves architectural, design, quality assurance, and management soft skills to be adopted and lived during a development process.

The context of this Opinion column is the inception of a new special interest group on RSE within the German Association for Computer Science (https://GI.de). However, RSE research cannot be a national activity: international RSE Research collaboration is required (https://irser.github.io), for which we call with this Opinion column. We support initiatives to raise awareness of the role of funding practice in the sustainability of research software, and to improve that practice, such as the Amsterdam Declaration on Funding Research Software Sustainability (https://ADORE.software). A recent Dagstuhl seminar brought together software engineering researchers and research software engineers bridging knowledge gaps among these communities (www.dagstuhl.de/24161). Another option could be a new task force or similar on the topic of RSE within ACM SIGSOFT, the special interest group on software engineering, so that interested persons have a place to go and cooperate.

References

1. Chue Hong, N.P. et al. *Software Sustainability Institute Midterm Rev.* (2023). 10.5281/ZENODO.8205595
2. Chue Hong, N.P. et al. FAIR Principles for Research Software (FAIR4RS Principles). (2022); 10.15497/RDA00068
3. Hasselbring, W. et al. From FAIR research data toward FAIR and open research software. *IT—Information Technology 62* 1, (Feb. 2020); 10.1515/itit-2019-0040
4. Hasselbring, W. et al. Open source research software. *Computer 53*, 8 (Aug. 2020); 10.1109/mc.2020.2998235
5. Hasselbring, W. et al. Multi-Dimensional categorization of research software with examples. *Zenodo* (2024); 10.5281/zenodo.14082554
6. Heroux, M.A. Research software science: Expanding the impact of research software engineering. *Computing in Science & Engineering 24*, 6 (2022); 10.1109/mcse.2023.3260475
7. Johanson, A. and Hasselbring, W. Software engineering for computational science: Past, present, future. *Computing in Science & Engineering 20*, 2 (Mar. 2018); 10.1109/mcse.2018.021651343
8. Kanewala, U. and Bieman, J.M. Testing scientific software: A systematic literature review. *Information and Software Technology 56*, 10 (Oct. 2014); 10.1016/j.infsof.2014.05.006
9. Kelly, D.F. A software chasm: Software engineering and scientific computing. *IEEE Software 24*, 6 (Nov. 2007); 10.1109/ms.2007.155
10. Lamprecht, A.L. et al. Towards FAIR principles for research software. *Data Science 3*, 1 (June 2020); 10.3233/ds-190026
11. Lamprecht, A.L. et al. What do we (not) know about research software engineering?. *J. Open Research Software 10* (Dec. 2022); 10.5334/jors.384
12. Li, Y. et al. Automated requirements extraction for scientific software. *Procedia Computer Science 51*, (2015); 10.1016/j.procs.2015.05.326
13. Ralph, P. et al. *Empirical Standards for Software Engineering Research.* (2021); 10.48550/arXiv.2010.03525 Version 0.2.0.
14. Randell, B. *Fifty Years of Software Engineering—or—The View from Garmisch.* (2018); 10.48550/arXiv.1805.02742
15. Reussner, R. et al. *Managed Software Evolution.* Springer, Cham (2019); 10.1007/978-3-030-13499-0

Michael Felderer (Michael.Felderer@dlr.de) is a professor in the Department of Computer Science at the University of Cologne and director of the Institute for Software Technology, German Aerospace Center (DLR), Germany.

Michael Goedicke (michael.goedicke@paluno.uni-due.de) is a professor emeritus for the specification of software systems in the Department of Computer Science at the University of Duisburg-Essen, Germany.

Lars Grunske (grunske@informatik.hu-berlin.de) is a professor of software engineering in the Department of Computer Science at Humboldt-Universität zu Berlin, Germany.

Wilhelm Hasselbring (hasselbring@email.uni-kiel.de) is a professor of software engineering in the Department of Computer Science at Kiel University, Germany.

Anna-Lena Lamprecht (anna-lena.lamprecht@uni-potsdam.de) is a professor of software engineering in the Department of Computer Science at the University of Potsdam, Germany.

Bernhard Rumpe (rumpe@se-rwth.de) is a professor of software engineering in the Department of Computer Science at RWTH Aachen University, Germany.

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/investigating-rse