Toward Research Software Engineering Research

MICHAEL FELDERER, German Aerospace Center (DLR), Germany
MICHAEL GOEDICKE, University of Duisburg-Essen, Germany
LARS GRUNSKE, Humboldt-Universität zu Berlin, Germany
WILHELM HASSELBRING, Kiel University, Germany
ANNA-LENA LAMPRECHT, University of Potsdam, Germany
BERNHARD RUMPE, RWTH Aachen University, Germany

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

Research Software Engineering (RSE) and the related role as Research Software Engineer has emerged as a job profile in its own right. We propose 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. Context of this viewpoint is the emerging National Research Data Infrastructure Germany (NFDI.de), where RSE is an important cross-cutting topic, and the inception of a new special interest group on RSE within the German Association for Computer Science (GI.de).

To set the stage, we start with a look at fifty years of software engineering research, and introduce the characteristics of research software, of RSE in general and RSE Research in particular. We conclude with an outlook to further essential activities on RSE Research.

1 SOFTWARE ENGINEERING RESEARCH

Randell [10] reviewed the first fifty 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 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 text books on software engineering, a multitude of books on dedicated sub-fields 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.

Authors' addresses: Michael Felderer, Michael.Felderer@dlr.de, German Aerospace Center (DLR), Cologne, Germany; Michael Goedicke, goedicke@informatik.uni-essen.de, University of Duisburg-Essen, Essen, Germany; Lars Grunske, grunske@informatik.hu-berlin.de, Humboldt-Universität zu Berlin, Berlin, Germany; Wilhelm Hasselbring, hasselbring@email.uni-kiel.de, Kiel University, Kiel, Germany; Anna-Lena Lamprecht, anna-lena.lamprecht@uni-potsdam.de, University of Potsdam, Potsdam, Germany; Bernhard Rumpe, rumpe@se-rwth.de, RWTH Aachen University, Aachen, Germany.

Software Engineering does not only have these sub-disciplines mentioned above, 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, as software engineering research largely ignored the specific demands of research software [7], and vice versa.

2 RESEARCH SOFTWARE

Research software is software that is designed and developed to support research activities in various fields such as science and humanities. It can be used to collect, process, analyze, and visualize data, as well as to model complex phenomena and run simulations. Research software can be 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 mainly falls into one of the following categories (and sometimes combinations):

- Modeling, simulation and data analytics of, e.g., physical, chemical, social, or biological processes in spatio-temporal contexts
  - Numerical and agent-based modeling and simulation (in silico experiments).
  - Data assimilation.
  - Data science and data engineering.
- Embedded control software for complex physical or chemical experiments and instruments, including many forms of sensor-based data collection.
- Proof-of-concept software prototypes in engineering research.
- Infrastructure and platform software, such as research data and software management systems.

Research software includes source code files, algorithms, scripts, computational workflows, and executables that are created during the research process or for a research purpose. Software components (e.g., operating systems, programming languages, libraries, etc.) that are used for research but were not created during or with a clear research intent should be considered 'software in research' and not 'research software' [1]. Research software should be FAIR [3, 8] 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.

3 RESEARCH SOFTWARE ENGINEERING

Research software engineering (RSE) 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 research software engineering 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 PhD 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 [6].

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 UK Society of Research Software Engineering (society-rse.org), the US Research Software Engineer Association (us-rse.org), or the German Society for Research Software (de-rse.org), which in turn coordinate in an international council (researchsoftware.org/council.html). The UK Software Sustainability Institute (www.software.ac.uk) provides support for furthering RSE practice, while the Research Software Alliance (ReSA, researchsoft.org) works towards evolving RSE policy by collaborating with decision makers and key stakeholders. We propose to complement these activities via *RSE Research*.

4 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 has to 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 (cf. github.com/NLeSC/RSE-research for a larger collection):

- What is specific about RSE compared to other SE specializations?
- How to (better) organize software-centric scientific processes?
- How to (better) integrate SE techniques like requirements engineering, architectural modelling 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?

We acknowledge that Heroux [5] 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.

5 CONCLUSIONS 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 [6]. Research software engineering correctly pushes these techniques into development projects for researchers, but domain-specific adaptations are necessary. This way, research may achieve maintainable, long-living software [2], in particular for community research software. We suggest addressing this goal via empirical software engineering research [9].

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, etc.

Researchers need to 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 need to 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, etc. It is necessary to build better domain-specific tooling to address the domain-specific challenges of research software. This brings us to establish RSE Research as a research field over RSE.

Research requires appropriate funding, which will also be required for RSE Research. For Germany, we consider proposing a new DFG (German Research Foundation) priority program, similar to a previous program for long-living software systems [2]. However, RSE Research cannot be a national activity: international RSE Research collaboration is required, for which we call with this viewpoint article. 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 (ADORE.software).

REFERENCES

[1] Neil P. Chue Hong et al. 2022. FAIR Principles for Research Software (FAIR4RS Principles). https://doi.org/10.15497/RDA00068

[2] Ursula Goltz, Ralf H. Reussner, Michael Goedicke, Wilhelm Hasselbring, Lukas Märtin, and Birgit Vogel-Heuser. 2015. Design for future: managed software evolution – The DFG priority programme for long-living software systems. *Computer Science – Research and Development* 30, 3-4 (Oct. 2015), 321–331. https://doi.org/10.1007/s00450-014-0273-9

[3] Wilhelm Hasselbring, Leslie Carr, Simon Hettrick, Heather Packer, and Thanassis Tiropanis. 2020. From FAIR research data toward FAIR and open research software. *it - Information Technology* 62, 1 (Feb. 2020), 39–47. https://doi.org/10.1515/itit-2019-0040

[4] Wilhelm Hasselbring, Leslie Carr, Simon Hettrick, Heather Packer, and Thanassis Tiropanis. 2020. Open Source Research Software. *Computer* 53, 8 (Aug. 2020), 84–88. https://doi.org/10.1109/mc.2020.2998235

[5] Michael A. Heroux. 2023. Research Software Science: Expanding the Impact of Research Software Engineering. *Computing in Science & Engineering* (2023), 1–7. https://doi.org/10.1109/mcse.2023.3260475

[6] Arne Johanson and Wilhelm Hasselbring. 2018. Software Engineering for Computational Science: Past, Present, Future. *Computing in Science & Engineering* 20, 2 (March 2018), 90–109. https://doi.org/10.1109/mcse.2018.021651343

[7] Diane F. Kelly. 2007. A Software Chasm: Software Engineering and Scientific Computing. *IEEE Software* 24, 6 (Nov. 2007), 120–119. https://doi.org/10.1109/ms.2007.155

[8] Anna-Lena Lamprecht et al. 2020. Towards FAIR principles for research software. *Data Science* 3, 1 (June 2020), 37–59. https://doi.org/10.3233/ds-190026
[9] Paul Ralph et al. 2021. Empirical Standards for Software Engineering Research. https://doi.org/10.48550/arXiv.2010.03525 Version 0.2.0.
[10] Brian Randell. 2018. Fifty Years of Software Engineering – or – The View from Garmisch. https://doi.org/10.48550/arXiv.1805.02742

Michael Felderer (Michael.Felderer@dlr.de) is a full 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 (goedicke@informatik.uni-essen.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 full 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 full 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 full professor of Software Engineering in the Department of Computer Science at the University of Potsdam, Germany.
Bernhard Rumpe (rumpe@se-rwth.de) is a full professor of Software Engineering in the Department of Computer Science at RWTH Aachen University, Germany.