Electronic Communications of the EASST
Volume 83 Year 2025

deRSE24 - Selected Contributions of the 4th Conference for Research Software Engineering in Germany

Edited by: Jan Bernoth, Florian Goth, Anna-Lena Lamprecht and Jan Linxweiler

Compared Experiences from Teaching Full-Semester Research Software Engineering Courses at Four German Universities

Nikolas Bertrand, Akshay Devkate, Guido Juckeland, Jan Linxweiler, Sören Peters, Steffen Remus, Katrin Schöning-Stierand, Anna-Lena Lamprecht

DOI: 10.14279/eceasst.v83.2611

Electronic Communications of the EASST (https://eceasst.org).
Published by **Berlin Universities Publishing**
(https://www.berlin-universities-publishing.de/)

Compared Experiences from Teaching Full-Semester Research Software Engineering Courses at Four German Universities

Nikolas Bertrand[1], Akshay Devkate[2], Guido Juckeland[3], Jan Linxweiler[4], Sören Peters[5], Steffen Remus[6], Katrin Schöning-Stierand[7], and Anna-Lena Lamprecht[8]

[1] University of Potsdam, nbertrand@uni-potsdam.de,
[2] University of Potsdam, akshay.devkate@uni-potsdam.de
[3] Helmholtz-Zentrum Dresden - Rossendorf / TU Dresden, g.juckeland@hzdr.de
[4] TU Braunschweig, j.linxweiler@tu-braunschweig.de
[5] TU Braunschweig, soe.peters@tu-braunschweig.de
[6] Universität Hamburg, steffen.remus@uni-hamburg.de
[7] Universität Hamburg, katrin.schoening-stierand@uni-hamburg.de
[8] University of Potsdam, anna-lena.lamprecht@uni-potsdam.de

Abstract: Research Software Engineering has emerged as a critical discipline at the intersection of software development and research with the aim of enhancing the quality, reliability, and reproducibility of scientific software. In this paper we present a comparative analysis of the experiences gained from teaching full-semester Research Software Engineering (RSE) courses at four different universities in Germany. Despite its growing importance, there is limited literature on the pedagogical approaches and challenges encountered in teaching RSE courses, particularly at the university level. This paper investigates and contrasts the contexts, designs, and experiences of RSE courses offered at the TU Braunschweig, TU Dresden, University of Hamburg and University of Potsdam. By synthesizing the experiences and insights gleaned from these four universities, this study aims to provide valuable guidance and best practices for educators seeking to develop or enhance RSE education initiatives.

Keywords: Research Software Engineering, Scientific Software Engineering, RSE Education, RSE Teaching

1 Introduction

Research Software Engineering (RSE) skills have become increasingly important across all scientific disciplines, as modern research heavily relies on software tools and computational methods. This necessitates the development of suitable educational and training forms to equip researchers with these essential skills. Currently, there is a vibrant and ongoing discussion within the RSE communities about the best approaches to teaching and training in this field.

For example, Barker et al. [BBB+24] highlight this need in their report on "Software and skills for research computing in the UK", underscoring the importance of well-defined training pathways. In Germany, this discourse has been furthered by the "Teaching RSE" working group,

which formed at the 2023 German RSE conference (deRSE23) in Paderborn. This group has recently released a paper titled "Foundational Competencies and Responsibilities of a Research Software Engineer" [GAB+24] as the basis for further discussion and developments. Among other recommendations, the paper advocates for the creation of a dedicated RSE Master's program, a proposal that has garnered significant interest and led to the formation of a separate working group. Additionally, resources such as the website https://de-rse.org/learn-and-teach/ created by the group serve as portals and link collections, providing valuable materials related to RSE teaching. Among many other things, they point to the Software Carpentry (https://software-carpentry.org/) [Wil16], a community of volunteer instructors who teach short workshops and develop lessons about various RSE skills. They have proven very effective in filling skill gaps on an as-needed basis and are particularly popular among researchers (PhD students, postdocs) who look to enhance specific technical skills.

This paper focuses on teaching full-semester RSE courses at universities that provide opportunities for students to develop substantial RSE skills already during their studies. Naturally, the design of these courses must be tailored to their specific organizational contexts, considering various factors such as institutional resources, student backgrounds, and educational goals. We explore four different RSE courses taught at four German universities: TU Braunschweig (TUB), TU Dresden (TUD), University of Hamburg (UHH), and University of Potsdam (UP). The compared experiences reported here are based on a series of discussion sessions among the authors, which started at the deRSE24 Conference in Würzburg in March 2024 as a result of related contributions, were continued at the Dagstuhl seminar "Research Software Engineering: Bridging Knowledge Gaps" [DGJK24] in April 2024, and eventually resulted in the collaborative work on this report. By sharing our experiences, we aim to provide inspiration and guidance for others looking to implement similar programs, and to contribute to the broader conversation on effective RSE education, helping to shape the future of training in this field.

The paper is structured as follows: Section 2 provides an overview of the contexts and origins of the four RSE courses. Section 3 compares the course designs, focusing on content, formats, learning outcomes, and assessment methods. Section 4 discusses experiences with the different courses, incorporating student feedback and observations, and adjustments made in response. Section 5 concludes the paper, summarizing key insights and discussing future directions for RSE education at universities.

2 Course Contexts

The courses at TU Braunschweig, TU Dresden, University of Hamburg, and the University of Potsdam exhibit distinct characteristics shaped by their institutional contexts. For orientation, we start with an overview of the contexts and origins of the four RSE courses discussed in this paper, with Table 1 summarizing key points.

At TU Braunschweig the RSE course goes by the name "Scientific Software Engineering" (SSE). It was first started in the Summer of 2022 and is a core component of the international, interdisciplinary, and research-oriented Master's degree program "Computational Sciences in Engineering (CSE)." Successful attendance in the course is compulsory for these students. However, the course is also open as an elective to students from other disciplines who want to spe-

Organization	TU Braunschweig	TU Dresden	U Hamburg	U Potsdam
Teaching team, partners	Institute of Computational Modeling in Civil Engineering	Institute of Software- and Multimedia-Technology with a teaching assignment to Computational Science Department of HZDR	Staff members of the House of Computing and Data Science & the Language Technology group	Chair of Software Engineering
Course title	Scientific Software Engineering	Introduction to Research Software Engineering	Research Software Engineering	Research Software Engineering
Contact hours/week	4 (2 lecture, 2 lab)	2 (1 lecture, 1 lab)	4 (2 lecture, 2 lab)	4 (2 lecture, 2 lab)
Credit points	5 ECTS	3 ECTS	6 ECTS	6 ECTS
Compulsory or elective	compulsory	elective	elective	elective
Related courses in curriculum	Algorithms and Data Structures; Parallel and Distributed Computing	Software Engineering	No related courses	Software Engineering, Research Data Management
Target audience (study programs)	Master students from Computational Sciences in Engineering (CSE) and (starting in 2026) Artificial Intelligence for Molecular Sciences (AIMS)	Open to all students with computer science affiliation (currently mostly Master students Computational Modeling and Simulation (CMS) and Computer Science Bachelor and Master students)	Open to all (Bachelor & Master)	Open to all (currently mostly Bachelor students from Cognitive Sciences, Computer Linguistics and Informatics, Master students from Computational Science, Data Science, and some other science programs)
Prerequisites	Programming skills (Python)	Programming skills (any language)	Programming skills (Python)	Programming skills (preferably Python)

Table 1: Course Contexts

cialize in scientific software development. The course was established in accordance with the digitization strategy of TU Braunschweig to reflect the growing importance of software in the engineering industry and academia. Hence, the course is meant to enable students to work at the intersection of traditional and software engineering disciplines. The course also served as a basis for workshops in SURESOFT [BDF+22], a project to support the sustainability of research software and will also be part of the Master's degree program "Artificial Intelligence for Molecular Sciences (AIMS)" beginning in 2025.

As Helmholtz-Zentrum Dresden - Rossendorf (HZDR) closely collaborates with TU Dresden, quite a number of students with a CS background also work in RSE roles at HZDR. The "Introduction to RSE" course at TU Dresden was started as a reaction to working with CS students either during their thesis or as student assistants and discovering that they need to be taught RSE fundamentals over and over again. The course aims at translating the experiences and fundamental skills of the RSE community both at HZDR and at TU Dresden to the students, also so that they can already be more productive during their academic studies. The large majority of students attending this course come from the Master program "Computational Modelling and Simulation". Students describe their motivation mainly as upskilling their software development skills as quite a number of them have a non-CS background and previously were more of the category "scientists who code" than actual software developers.

At University of Hamburg, the "Research Software Engineering" course was offered as a collaborative course by the House of Computing and Data Science (HCDS, lectures) and the Language Technology Group (LT, practice class) for the first time in the summer term of 2024. As a central institution, the HCDS supports interdisciplinary research and the introduction of innovative digital methods. At the same time, it coordinates and supports the implementation of the digitization strategy in research at UHH. The RSE course is offered to both Bachelor's and Master's students. It is suitable for participants from all disciplines with basic knowledge of Python or a similar programming language. Its design largely follows the example of the RSE course at the University of Potsdam (see below). Participants come from diverse fields of study, such as Political Science, Economics, Physics, or Psychology. In addition to this course, there is be an interdisciplinary Python course in the winter semester to prepare as many students as possible to participate in the RSE course.

The "Research Software Engineering" course at University of Potsdam was introduced by the Software Engineering group in the summer term of 2023 mainly to cater for the need of research-specific software engineering courses in the Master programs Computational Science and Data Science. With only average Python programming skills as a prerequisite and no particular disciplinary focus, it is however suitable for interested students from all programs, and is principally open to all. Indeed, the backgrounds of participants turned out to be very diverse, with e.g. Bachelor students from informatics, business infomratics, cognitive sciences and computer linguistics, and Master students from astrophysics, computational science, climate science, data science, ecology, economics, and more. (Experiences with this heterogeneous audience are discussed in Section 4.) The course also complements the interdisciplinary "Research Data Management" course that had been introduced by the computer science institute some years earlier in response to the increasing need for data management skills across all disciplines.

3 Course Designs

Driven by their different institutional contexts, the course designs at TU Braunschweig, TU Dresden, University of Hamburg, and University of Potsdam differ significantly in their focus, content, and assessment methods. In this section we take a closer look at the designs of the four different courses, with Table 2 summarizing essential points.

The goal of the "Scientific Software Engineering" course at TU Braunschweig is to provide students with a solid understanding of software engineering concepts, enabling them to design and build software that is easy to maintain and evolves over time while addressing the ever-growing complexity of research software. As such, the course emphasizes well-established practices and covers the basics of object-oriented design, Software Design Principles, Software Design Patterns, and Test-Driven development. The weekly lectures mix theoretical and practical elements, with a focus on close interaction with the audience to find and develop solutions together. Additionally, at the end of the semester, all participants consolidate what they have learned in group work on a larger practical example project. The group work results are then presented as a prerequisite for the final exam.

At TU Dresden, the course is offered as an elective and open to anyone who can receive credits from the computer science faculty (which is in total seven degree programs) and greatly benefits from the multitude of backgrounds of the students. The course is mainly programming language-agnostic and focuses on the software development processes in highly interdisciplinary team that requires computer scientists to negotiate good enough practices with scientists who code. It tries to follow a typical RSE consulting project life cycle from the RSE team at HZDR, from the initial building of a common vocabulary among interdisciplinary teams and also covering requirement analysis, working with software project platforms, testing/CI/CD, build systems, licensing and software publication. It uses both the training material of the Helmholtz RSE platform HIFIS (https://hifis.net/services/software/training.html) and the MIT "missing semester in CS education" (https://missing.csail.mit.edu/) as its content basis.

At University of Hamburg, the RSE course comprises a lecture and a hands-on part. Following the 2023 course design of the University of Potsdam, students are engaged to perform reusable data analysis as part of an individual research software project based on self-selected data in the first half of the course. In the second half of the course, students are engaged to perform a similar study in a group project, hence promoting teamwork and encouraging them to fulfill different user roles. In addition to existing fundamental programming skills, the necessary knowledge was imparted in the lecture. This included the use of Git, project structuring, Python code style, software licensing, and the use of Jupyter Notebooks. The content is mainly based on parts of the textbook "Research Software Engineering with Python" [IHJ+21]. Depending on the subject, the teaching is performed through slides, live coding, or a mix of both.

The "Research Software Engineering" course at University of Potsdam aims to enable students to conduct small and medium-sized research software projects professionally, assuming average Python programming skills but no software project experience at the start. The content of the course is largely based on chapters from the textbook "Research Software Engineering with Python" [IHJ+21] that originated from the Software Carpentry community, but complements it with additional content around FAIR software [HKB+22], the software life cycle (requirements, architectures, design) and workflow management with, e.g., Snakemake [MJL+21].

Organization	TU Braunschweig	TU Dresden	U Hamburg	U Potsdam
Main learning objective	Work at the boundary of classical engineering teams and software development teams	How to work in interdisciplinary teams on research software	Mainly data analyis with self-motivated objectives and hypotheses in two projects (individual & team)	Conduct small and medium-sized research software projects professionally.
Content/ topics	Dealing with the growing complexity of research software, Developing software which is easy to change, Object-Oriented Design, Design Principles, Design Patterns, Test-Driven Development	Working in teams, CI/CD, build systems, licensing, software publication	Project organization, coding style, version control, licensing, software citation, FAIR, teamwork, software life cycle (requirements, architectures, design, implementation, testing, deployment), error handling, packaging, configuration, workflow management.	
Basis/ textbook	Based on "Agile Software Development, Principles, Patterns, and Practices" [Mar13]	HIFIS Trainings (https://hifis.net/services/software/training.html), MIT Missing Semester of CS Education (https://missing.csail.mit.edu/)	"Research Software Engineering with Python" [IHJ+21], supplemented with own material on FAIR software, the software life cycle, and workflow management.	
Course format	Mix of theoretical and practical parts integrated into the weekly lectures	Following a RSE project life cycle from requirements gathering to software publication	Lecture with slides, jupyter notebooks, live coding, shared experiences, guest talks, with optional exercises to deepen the knowledge and the closely supervised project work, which will be used for grading	Centered around two course projects. Lectures and labs gradually introduce the required knowledge and skills.
Tools used	GitLab, VSCode, VSCode-Live-Share, Python, MyPy	GitLab, HedgeDoc, Zenodo	GitHub/GitLab, Jupyter Notebook, JupyterHub, IDE and programming language of choice (mainly Python, some R), and VSCode as IDE, Docker	GitLab, Python, Jupyter Notebook and IDE, text editor of choice and snakemake
Assessment	Group work on a larger practical example with presentation as a prerequisite for admission to the exam, and written exam	Exam (written or oral)	Individual project as a prerequisite for examination (rated on a binary scale), 50% group project + presentation, 50% examination (oral)	20% individual project (computational narrative), 40% group project (workflow application), 40% exam.

Table 2: Course Designs

Organization	TU Braunschweig	TU Dresden	U Hamburg	U Potsdam
Number of participants	2023: ca. 50 2024: ca. 30	2023: ca. 20 2024: ca. 65	2023: - 2024: ca. 20	2023: ca. 60 2024: ca. 90
Student feedback	Students enjoy the mix of theory and practice during lectures.	Positive, especially from CMS students that sometimes have a non-CS background	Positive. Students state that the course increased their interest in the studies and motivated them to engage with the content beyond the course.	Very positive. Students (also those from CS!) emphasize the great practical relevance and usefulness of the new skills acquired.
Major changes between years	No major changes, but the practical examples will be continuously improved based on feedback from participants	None yet	n/a	2024: Stronger focus on workflow management, in lecture and group project.
Best experience	Starting in 2026 the course will be part of the program "Artificial Intelligence for Molecular Sciences (AIMS)"	Already asked for exporting this course to another faculty	The diverse backgrounds of participants lead to a great variety in project topics; their own research ideas keep them motivated	Really cool interdisciplinary projects by mixed groups of students.
Greatest pain point	Balance between practical application and a methodical approach in the examples	Too little time allocated so far to make the lessons stick. Inflexible exam regulations to credit practical work of the students.	None yet	Inflexible study and examination regulations, making it difficult to recognize credits in other study programs, thus discouraging interested students from participating.

Table 3: Experiences

Students carry out two accompanying projects during the course that form the major part of the assessment: Students start with creating their own project on an individual basis, where they develop a Jupyter notebook-based computational narrative around a self-selected dataset from the German federal statistical office (Destatis). After that, the students continue with a group project. Here they have to develop a workflow application using Snakemake for a self-selected data analysis problem (typically proposed by a non-CS student in the group). In both projects, they are instructed to follow the RSE practices discussed in the lectures and labs. To encourage further reuse of the course material, both the lecture content in form of an online textbook (https://software-engineering-group-up.github.io/RSE-UP) and the course assignments (https://gitup.uni-potsdam.de/seg/rse_course/rse_course_materials) are shared under a Creative Commons Attributions (CC-BY) license, with associated code being shared under a MIT license.

4 Experiences

The RSE courses at TU Braunschweig, TU Dresden, University of Hamburg, and University of Potsdam offer varied experiences and insights, with each institution facing unique challenges and successes. In this section we describe and compare our experiences with teaching the different RSE courses, ranging from first-time experiences like in Hamburg to accounts from several years of teaching and further developing the course like in Braunschweig. Table 3 provides an overview of key experiences and insights.

The "Scientific Software Engineering" course at TU Braunschweig has been well accepted, and students provided positive and valuable feedback. The students appreciate the blend of theoretical and practical content. In particular, the informal interactive part has been appreciated, as it supports flexible discussion of topics and questions. Also, the opportunity to deepen their understanding through homework assignments and hands-on projects has been very well received. Additionally, students value the collaborative environment fostered by group work and the chance to present their projects, consolidating their knowledge and enhancing their communication and teamwork skills. In addition, they value that what they learned from this course can be applied to a wide range of other subjects and thesis projects.

The RSE course at TU Dresden saw a huge increase of students to 65 in the second year and has also been approached about an export to other faculties. There has already been a request to also offer this course to students of another faculty to support their domain-specific data and information science-centered master program. However, in its current format the course offers too little time for students to consolidate the learned concepts so that they can flexibly adapt them in their future academic or non-academic career. It is therefore planned to extend the course with a weekly project work block, as this is reported as highly beneficial from the other three partners of this paper.

Although the first iteration of the RSE course at the University of Hamburg had only around 20 participants, it was still very diverse: the participants came from more than 10 different Bachelor's and Master's programs. This led to a good starting situation in implementing the group projects and the option to work in an interdisciplinary team. The small course size enables good contact with the participants and direct feedback from them. Students were generally very happy with the course. Comments in the evaluation indicated that the course was particularly suitable for those with very little prior knowledge in the field. For the upcoming course, the focus will be on promoting it in advance to reach more students. In this regard, a Python course in the winter semester for students and researchers, organized by the HCDS, will also be beneficial.

In Potsdam, the RSE course started in 2023 with around 60 students in the first iteration – already three times more than anticipated (luckily additional staff was available to support an upscaling at short notice) – and saw an increase to around 90 participants in the second round. Although being an elective, it is likely that the number will increase further in the coming years, as more study programs discover the course and recognize the credits for their students. As mentioned before, the participants in the course are very diverse, coming from more than ten different Bachelor and Master programs. While the diverse backgrounds can obviously be challenging, the course design is not just explicitly domain-agnostic, but rather takes advantage of this situation and lets students from computer/computational science, data science and other disciplines work together in one group, so that they benefit and learn from the respective specific

knowledge and skills of their teammates.

Our discussion of experiences also reflected that effective teaching requires not only a deep understanding of the subject matter, but also a keen awareness of best practices in classroom management and student engagement. Below we list the most important "do's and don'ts" from our experience, providing guidance to some key aspects of successfully teaching RSE courses:

Do:

- ... **Carpentry-style [Wil16] interactive coding sessions** when suitable for the content. Slides are often not the most suitable medium for explaining code and similar artifacts. Live coding with interactive discussion makes it more accessible to students.
- ... **let students work onpractical projects**, ideally building on their own ideas. As RSE is a practical topic, good RSE courses need to have a strong practical component. Problem-based learning generally increases student motivation and dedication, and we experienced that letting students (partially) define their own projects increases their engagement further.
- ... **encourage team work** (e.g. group projects, students reviewing other students merge requests, collaborative coding). While teamwork in student groups always comes with challenges, it is of utmost importance in RSE, and thus needs to be trained in an RSE course.
- ... **recap content** of the last lecture to refresh memory. Briefly reminding students of last time's topics is generally a good idea at the start of a lecture. We felt that for RSE, which is for many students quite orthogonal to their main study discipline, this is very important to get (back) on track before discussing new topics.
- **... provide lab exercises of different difficulty** to cater for all skill levels of the students. The great diversity in the participants' backgrounds also means that some need a lot of help while others are quickly done and bored. We saw that it paid off to also pose some more challenging extra assignments as well as extra refresher exercises regularly, or motivate the stronger students to help their peers.
- ... **use collaborative notes** (also for immediate feedback after each lecture/lab). Collaborative note-taking (e.g. by using an EtherPad) can help students to stay focused, enables low-key interaction and can in our experience lead to more questions being asked and discussions among students, already during the lecture.

Don't:

- ... **expect anything beyond programming skills** as prior knowledge. Most students signing up for the RSE courses will have just done a basic programming course, probably in Python. Other skills, like version control or shell scripting, are sometimes there, but should in our experience not be taken for granted.
- ... **use examples with too complex domain problems**. Instead, use problems that everyone is familiar with or can understand easily. Such examples can sometimes be hard

to find, but in our experiences it is worth the time investigating, as the suitability of the examples greatly influences student motivation and engagement.

- ... **underestimate the interest in the course** and prepare to scale up when more students show up. Almost all of us experienced a larger interest in the course than anticipated. We will have to see if the upward trend continues in the future, but we will certainly prepare for scaling up if needed.

5 Conclusion

The comparative analysis of full-semester Research Software Engineering ourses at TU Braunschweig, TU Dresden, University of Hamburg, and University of Potsdam reveals valuable insights into the diverse approaches and experiences of teaching RSE within the German higher education context. Each course is tailored to the specific needs and backgrounds of their student cohorts, reflecting the unique institutional contexts and goals. Despite the differences in course designs and institutional contexts, common themes emerge. A balanced approach between theoretical foundations and practical applications is key. Student feedback across all institutions underscores the value of practical, project-based learning and the benefits of interdisciplinary collaboration. However, challenges such as inflexible examination regulations and the need for continuous improvement in course content and delivery are also prevalent. Addressing these challenges requires ongoing adaptation and feedback-driven modifications to ensure that the courses remain relevant and effective.

The experiences shared in this paper contribute to the broader conversation on RSE education, offering practical guidance for developing and implementing similar programs at other institutions. Future directions for RSE education should further promote interdisciplinary collaboration, and expand the reach of these courses through effective promotion and by offering preparatory courses, such as the Python course at UHH, to help attract a broader student base.

Bibliography

[BBB+24] M. Barker, E. Breitmoser, P. Broadbent, N. Chue Hong, S. Hettrick, I. Lampaki, A. Quinn, R. Taylor. Software and skills for research computing in the UK. Technical report, Zenodo, Jan. 2024.
doi:10.5281/zenodo.10473186
https://zenodo.org/records/10473186

[BDF+22] C. Blech, N. Dreyer, M. Friebel, C. Jacob, M. Shamil Jassim, L. Jehl, R. Kapitza, M. Krafczyk, T. Kürner, S. C. Langer, J. Linxweiler, M. Mahhouk, S. Marcus, I. Messadi, S. Peters, J.-M. Pilawa, H. K. Sreekumar, R. Strötgen, K. Stump, A. Vogel, M. Wolter. SURESOFT: Towards Sustainable Research Software. 2022.
doi:10.24355/dbbs.084-202210121528-0
https://leopard.tu-braunschweig.de/receive/dbbs_mods_00071451

[DGJK24] S. Druskat, L. Grunske, C. Jay, D. S. Katz. Research Software Engineering: Bridging Knowledge Gaps (Dagstuhl Seminar 24161). *Dagstuhl Reports* 14(4):42–53, 2024. doi:10.4230/DagRep.14.4.42 https://drops.dagstuhl.de/entities/document/10.4230/DagRep.14.4.42

[GAB+24] F. Goth, R. Alves, M. Braun, L. Castro, G. Chourdakis, S. Christ, J. Cohen, S. Druskat, F. Erxleben, J. Grad, M. Hagdorn, T. Hodges, G. Juckeland, D. Kempf, A. Lamprecht, J. Linxweiler, F. Lffler, M. Martone, M. Schwarzmeier, H. Seibold, J. Thiele, H. von Waldow, S. Wittke. Foundational Competencies and Responsibilities of a Research Software Engineer [version 1; peer review: 1 approved]. 2024. doi:10.12688/f1000research.157778.1

[HKB+22] N. Hong, D. Katz, M. Barker, A.-L. Lamprecht, C. Martinez, F. Psomopoulos, J. Harrow, L. Castro, M. Gruenpeter, P. Martinez, T. Honeyman, A. Struck, A. Lee, A. Loewe, B. Werkhoven, D. Garijo, E. Plomp, F. Genova, H. Shanahan, M. Hellström, M. Sandström, M. Sinha, M. Kuzak, P. Herterich, S. Islam, S.-A. Sansone, T. Pollard, U. Atmojo, A. Williams, A. Czerniak, A. Niehues, A. Fouilloux, B. Desinghu, C. Goble, C. Richard, C. Gray, C. Erdmann, D. Nüst, D. Tartarini, E. Ranguelova, H. Anzt, I. Todorov, J. McNally, J. Burnett, J. Garrido-Sánchez, K. Belhajjame, L. Sesink, L. Hwang, M. Tovani-Palone, M. Wilkinson, M. Servillat, M. Liffers, M. Fox, N. Miljković, N. Lynch, P. Lavanchy, S. Gesing, S. Stevens, S. Cuesta, S. Peroni, S. Soiland-Reyes, T. Bakker, T. Rabemanantsoa, V. Sochat, Y. Yehudi, FAIR4RS WG. *FAIR Principles for Research Software (FAIR4RS Principles)*. Mar. 2022. doi:10.15497/RDA00065

[IHJ+21] D. Irving, K. Hertweck, L. Johnston, J. Ostblom, C. Wickham, G. Wilson. *Research Software Engineering with Python: Building Software that Makes Research Possible*. CRC Press/Taylor and Francis, 2021. doi:10.1201/9781003143482

[Mar13] R. Martin. *Agile Software Development, Principles, Patterns, and Practices* . Pearson Deutschland, 2013. https://elibrary.pearson.de/book/99.150005/9781292038360

[MJL+21] F. Mölder, K. P. Jablonski, B. Letcher, M. B. Hall, C. H. Tomkins-Tinch, V. Sochat, J. Forster, S. Lee, S. O. Twardziok, A. Kanitz et al. Sustainable data analysis with Snakemake. *F1000Research* 10, 2021. doi:10.12688/f1000research.29032.2

[Wil16] G. Wilson. Software Carpentry: lessons learned. *F1000Research* 3:62, Jan. 2016. doi:10.12688/f1000research.3-62.v2