Foundational Competencies of a Research Software Engineer - The summary

The teachingRSE project [1]

[1] https://github.com/the-teachingRSE-project

2025-10-27

Abstract: The term Research Software Engineer, or RSE, emerged a little over 10 years ago as a way to represent individuals working in the research community but focusing on software development. The term has been widely adopted and there are a number of high-level definitions of what an RSE is. However, the roles of RSEs vary depending on the institutional context they work in. At one end of the spectrum, RSE roles may look similar to a traditional research role. At the other extreme, they resemble that of a software engineer in industry. Most RSE roles inhabit the space between these two extremes. Therefore, providing a straightforward, comprehensive definition of what an RSE does and what experience, skills and competencies are required to become one is challenging. In this community paper (a summary of a more extensive publication) we define the broad notion of what an RSE is, explore the different types of work they undertake, and define a list of foundational competencies as well as values that outline the general profile of an RSE. Further research and training can build upon this foundation of skills and focus on various aspects in greater detail, and we expect that graduates and practitioners will have a larger and more diverse set of skills than outlined here.

Keywords: research software engineering, RSE, competencies, curriculum design, teaching

Contents

1 Introduction

Research software is now used and developed not only in science, technology, engineering and mathematics (STEM) domains, but also in other fields, like medicine and the humanities. Researchers, however, often lack the skills to use specialised software for their research, let alone write it. If they come from a non-technical domain, they may also struggle to know what to ask when trying to request help from and interact with more experienced colleagues. A digital literacy gap still exists in academic education, which students and researchers have to fill by themselves.

Researchers investing increasing amounts of their time developing their software engineering (SE) skills to support their research work can find themselves with little time to do the research itself. This, in turn, presents career development challenges since the experience required to gain and progress in research and academic roles is traditionally assessed through metrics that do not directly include software outputs. A recent shift towards the establishment of the distinct role of a *"Research Software Engineer"* (RSE, a term that was coined in the United Kingdom (UK) in the early 2010s [7]), now provides a base on which sustainable career opportunities can be built, allowing for better training of researchers and more effective support for the development of high quality research software. Regardless of their explicit or implicit job title and the environment in which they work, RSEs share a set of core skills that are required to design and develop research software, understand the research environment, and ensure that they produce sustainable, maintainable code that supports reproducible research outputs, following the Findability, Accessibility, Interoperability and Reusability (FAIR) principles [1].

This community paper defines a initial set of core values (Section 2) and foundational competencies (Section 3), which an RSE should acquire. These are independent of a research domain and are drawn upon skills from traditional SE practice, established research culture, and the commitment to being part of a team. While being the result of workshop discussions that were attended largely by RSEs (deRSE23 in Paderborn, un-deRSE23 in Jena, and deRSE24 in Würzburg, all in Germany), we believe that these competencies can offer far-reaching impact beyond the domain of RSE into the wider research community. This is especially important given that much research involves some amount of data management, processing and visualisation, or the creation of tools for these tasks, and funding bodies and computing infrastructure providers will sometimes prioritise projects that generate archived, annotated, re-usable, and potentially remotely executable data.

This paper is a condensed version of a more comprehensive paper [5] published at [4], aimed as a call to a broader audience for further discussion and coordinated action. Besides the foundational competencies, that paper further elaborates on the guiding values and principles of an RSE, points to related work, and describes the tasks and typical level of each competency for different career levels and working environment, together with actions that organizations can take to support these.

1.1 Terminology

Depending on the national research environments and processes that readers are familiar with, the notion of the terms *software* and *research* might differ. Therefore, to avoid ambiguities, we define these as follows:

Software: Source code, documentation, tests, executables and all other artefacts that are created during the development process that are necessary to understand its purpose.

Research software: Foundational algorithms, the software itself, as well as scripts and computational workflows that were created during the research process or for a research purpose, across all domains of research. This definition is broader than in [1].

Research software engineers: People who create or improve research software and/or the structures that the software interacts with in the computational environment of a research domain. They are highly skilled team members who may also choose to conduct their own research as part of their role. RSEs fall therefore somewhere on the spectrum between a researcher at one end and a software engineer at the other. Common to all of them is, that they need to be able to work in the research environment the software is used in, ideally at eye-level with native researchers. However, we also recognise that many RSEs have chosen specifically to focus on a technical role as an alternative to a traditional research role because they enjoy and wish to focus on the development of research software.

Researchers: People who are using the services provided by Research Software Engineers. This, on purpose, is a very broad definition and was chosen for easier readability.

2 Values

It is important that the activities of an RSE are guided by ethical values. In addition to the values for good scientific practice [3], RSEs also adhere to the SE Code of Ethics [6]. Central to that code is the RSE's obligation to commit to the health, safety and welfare of the public and act in the interest of society, their employer and their clients. Further values loosely based on that code include the obligations

- to commit to objectivity and fact-based, honest research conclusions,
- to promote openness and accountability in the research process,
- to take great care to develop software that adheres to current best practices,
- to judge independently and maintain professional integrity,
- to treat colleagues and collaborators with respect and work towards a fair and inclusive environment, and
- to promote these values whenever possible and make sure that they are passed on to new practitioners.

The deployment of computer-based modelling and simulation has dramatically changed the practice of science in a large number of fields. It has enabled the hitherto impossible study of new classes of problems, often replacing traditional experimentation and observation.

The relationship between initial state, inputs and final state of a computer simulation is "epistemically opaque" [8], in that not every step of the process is directly observable. The current trend of an increasing application of computationally irreducible systems, such as those based on artificial neural networks, further exacerbates this inherent limitation of explainability. An RSE usually takes a pivotal role in assessing this adequacy for purpose of a model as well as in characterising and communicating the domain of its legitimate application and its limits of interpretability. This role, together with the enormous reliance on modelling and simulation of scientific results, as well as real-world decision-making, places a large responsibility on the RSE. It is important that RSEs are aware of this responsibility and continuously improve their capabilities to live up to it.

Research software is also well on its way to being ever-present in data-driven research, in all research fields. It is not unusual for RSEs to support those more research data oriented efforts as well. Here, specifically, they closely interact with research data management professionals and practices by designing research software that is better able to adhere to the FAIR principles for research data, but also to follow similar rules for research software (FAIR4RS [1]). As such, they are then familiar with special requirements stemming from the field itself, e.g., in medical research, and with privacy related issues especially for personal data, e.g., for conducting surveys.

RSEs often assume a multifaceted role at the junction of research, SE and data management. They work with a varying and diverse set of colleagues that might include other developers, support unit staff and academics of different fields and all career stages. This situation yields a specific set of challenges RSEs should be aware of to consciously make ethically sound judgement calls. Below we list some example areas that highlight present-day challenges.

2.1 Current challenges

A lot of RSE work involves the manipulation or creation of data processing tools. We highlight that professional conduct requires these creations to be reliable and to maintain data integrity. In particular, the way that personal data is handled can have far-reaching implications for society. Independent of the encoding into the respective national law in an RSE's jurisdiction, the right to information privacy is internationally recognised as a fundamental human right.

RSEs are often experienced professionals who work closely with and provide technical training and guidance to early career researchers. Similarly to academic supervisors, they bear a certain responsibility to guide and advise less-experienced colleagues with respect to career development and the achievement of academic goals. This can take the form of supervising students or mentoring fellow RSEs. The RSE needs to be aware of the biases arising from the sociological imbalances in research and academia.

There is an ever-growing demand for resources to cover the expanding need of storage and processing, with no clear deceleration in sight. At the same time, current science is well aware of several planetary boundaries being exceeded due to human activities [10]. The Governance, Responsibility, Estimation, Energy and embodied impacts, New collaborations, Education and Research (GREENER) principles [9] suggest how these concerns can be addressed and how research computing can become more environmentally sustainable.

RSEs often operate at the cutting edge of technological development and therefore might have to deal with technologies of which the dangers and drawbacks are still poorly understood. A current example is the rush for the application of large language models (LLMs), where RSEs working in these fields should stay up-to-date and be able to help researchers assess topics such as training-data bias, LLM "hallucinations" or malicious use, with the greater goal of making these powerful tools work for the welfare of society.

3 Foundational RSE competencies

The role of an RSE lies somewhere on the spectrum between that of a researcher (the "R") and a software engineer (the "SE") and, therefore, requires competencies in both fields. RSEs typically have a background in research or software engineering, but they definitely have obtained broader knowledge in both fields. Even when working as the only RSE on a task or project, they typically apply their knowledge and experience as part of larger teams of researchers and technical professionals. There are many ways to categorise the competencies of an RSE. We chose to distribute these competencies over three pillars to reflect the fact that RSEs are both competent researchers (the research skills, Section 3.2) and software engineers (the software/technical skills, Section 3.1). The third pillar (communication skills, Section 3.3) forms the bridge between the former two categories, with a particular focus on the software and research cycle and the scientific process. These competencies are relevant in a broad setting and form the foundation for specific specialisations. These competencies have been chosen in order to make RSEs contribute to an open and inclusive research environment, with tools that respect their professional values.

During the Paderborn workshop (deRSE23) we asked learners and novice RSEs what they would like to have learnt. The top five items mentioned were: testing, contributing to large projects, when or why to keep repositories private, high-quality software development, and finding a community. Those topics comprise combinations of the skills and competencies defined below.

3.1 Software/Technical skills

Besides skilled researchers, RSEs are also competent software engineers. As such, they can solve complex software engineering problems and design software as a user-oriented, future-proof product. The technical skills required by an RSE overlap to a large extent with the common fundamental software engineering skills, but put greater emphasis on aspects related to achieving good scientific practice and to serving special needs of research software. In addition, a lot of RSEs are either self- or peer taught in these skills. These skills include requirements analysis, design, construction, testing, program analysis, and maintenance of software. On the other hand, RSEs also know how to make research software adhere to the FAIR principles [1], and how to achieve different levels of research software reusability, while they have deeper understanding of the scientific context around the research software projects they work on. To reflect this, the technical skills listed below complement competencies regarding the standard life cycle of software development (as summarised in subsubsection 3.1.1) with RSE-specific focus skills.

3.1.1 Classical software engineering skills

To summarise the vast range of the skills a software engineer is typically equipped with, we refer to the Guide to the Software Engineering Body of Knowledge (Bourque, Fairley, and IEEE Computer Society [2]). Because research software engineering is an interface discipline, RSEs are often stronger in topics more commonly encountered in research software contexts (e.g., mathematical and engineering foundations) than in other areas (e.g., software engineering economics). However, they bring a solid level of competence in all software engineering topics. Therefore, RSEs can set and analyse software requirements in the context of open-ended, question-driven research. They can design software so that it can sustainably grow, often in an environment of rapid turnover of contributors. They are competent in implementing solutions themselves in a wide range of technologies fit for different scientific applications. They can formulate and implement various types of tests, they can independently maintain software and automate operations of the integration and release process. They can provide working, scalable, and future-proof solutions in a professional context and with common project and software management techniques, adapted to the needs of the research environment. Finally, as people who have often gained significant research experience in a particular discipline, they combine the necessary foundations from their domain with software engineering skills to develop complex software.

3.1.2 Adapting to the software life cycle (SWLC)

The traditional software development life cycle defines the stages that form the process of building a piece of software. Initial development generally involves an analytic process where requirements and ideas are gathered and analysed (requirements engineering), followed by formulating a plan to fulfil them (design) that is finally turned into running code (implementation). This is accompanied by different measures of quality control (e.g., reviews, testing), validating and verifying that things work as expected and that they continue to do so when development progresses further. Depending on the software project, this can mean a simple "Think-before-you-do", or more elaborate and formal processes. Often the development cycles are executed iteratively and incrementally. The life cycle further includes periods of deployment, maintenance and further development (software evolution), as well as software retirement. To assess the current state and needs of the software, the RSE should be familiar with different maturity metrics. Additionally, the research software life cycle extends the traditional life cycle with software publication. The RSE should be aware of this life cycle and be able to predict and cater to the changing needs of a software project as it moves through the stages.

3.1.3 Creating documented code building blocks (DOCBB)

The RSE should be able to create building blocks from source code that are reusable. This ranges from simple libraries of functions up to complex architectures consisting of multiple software packages. An important part of enabling code reusability is the provision of sufficient information in the form of comments within code, documentation or other means. This is vital to ensure that developers and maintainers understand what a piece of software aims to do and how to enable others to use the provided functionality. This is primarily achieved through a "clean" implementation and enhanced by documentation. Documentation ranges from commenting code blocks to using documentation (building) tools. It should be written with consideration for the different audiences who may need it depending on their goals and expertise.

3.1.4 Building distributable software (DIST)

The RSE should be able to distribute their code on their domain/language specific distribution platforms. This almost always encompasses handling/documenting dependencies with other packages/libraries. It sometimes requires knowledge of using build or package management systems to enable interoperability with other projects. In terms of usability and needs of the user community the RSE should be able to decide whether a library or a framework is the right type of program to build and distribute.

3.1.5 Use software repositories (SWREPOS)

The RSE should be able to identify and use fitting public platforms to share the artefacts they have created and invite the public to scrutinise them in public reviews. These software repositories usually provide facilities for software development, which differentiate them from the domain repositories described later.

3.1.6 Software behaviour awareness and analysis (MOD)

We define this as a certain quality of analytical thinking that enables an RSE to form a mental model of a piece of software in a specific environment (program comprehension). An RSE should be able to make predictions about the behaviour of a program. This is a required skill for common tasks such as debugging, profiling, optimising, designing good tests, or predicting user interaction. Many tools exist to help with understanding and evaluating existing code, especially from a structural point of view. An RSE should understand their output and its implications. An important facet of this capability relates to information security. RSEs need to consider the safety and integrity of personal data and other sensitive information and make sure that they do not negatively impact the integrity of their institution's network and computing infrastructure.

3.2 Research skills

3.2.1 Conducting and leading research (NEW)

RSEs are curious and able to conduct research, both on research software engineering, and on their research-wise "home domain". Senior RSEs are also able to lead research. Since RSEs often operate in different research fields, they also gain their reputation from their effectiveness in interacting with researchers from the same or other domains. Therefore, some curiosity together with a broad overview of the research field is required, as this enables the RSE to learn new methods and algorithms directly from domain peers. Similarly, a broad overview of the field of SE research and the growing field of RSE research enables the RSE to learn, apply, and teach new methods and tools for improving the way they develop software. This curiosity, together with the ability to convert it into new ideas, is also reflected when an RSE is actively trying out new tools or discovering related literature from adjacent domains. Lifelong learning is then no longer just a phrase but becomes a motivation to work.

3.2.2 Understanding the research cycle (RC)

One of the key skills that RSEs have is their understanding of how research works. They embrace being part of a larger community which, despite friendly competition, shares the common goal of gaining knowledge to disseminate it. Thereby they know that they are part of a bigger undertaking that involves many other parties in- and outside their domain, and also that their software can be utilised at different stages of the research cycle by different people. They may be asked to contribute to the ethical and regulatory evaluation of a project to ensure integrity of the research performed therein. Like other researchers, RSEs are open to discussions and arguments beyond their own expertise and appreciate the underlying principles of good research, including publications, reviews and reproducibility.

3.2.3 Software re-use (SRU)

The re-use of existing assets such as libraries and pieces of code to improve efficiency and quality belongs to the fundamentals of software construction [2]. To discover software, RSEs rely on domain-specific knowledge and domain repositories, as well as research skills, discovering related software via software citations and metadata. To evaluate whether the artefacts to be re-used suit their needs, RSEs often need to consider the scientific context of their origin. For example, a paper that references the code under consideration might be crucial to validate its fitness for purpose or lack of suitability. Code that incorporates research-domain specific knowledge needs to be understood at a very detailed level and its re-use documented to meet standards of good research practice. Not only the technical compatibility needs to be understood and documented (programming languages, system interoperability), but also the underlying models and computational methods need to fit the purpose; this question often requires wider research skills and deeper understanding of the research domain at hand.

3.2.4 Software publication and citation (SP)

Another part of FAIR software is concerned with publishing new and derived works and making them available for re-use by the research community and the general public. RSEs need to have a basic understanding of common software licence types, including proprietary and open source licences and how “copyleft” and “permissive” open source licences differ. They should also understand compatibility between different licences, and the ramifications for re-using and composing programs. Beyond that, RSEs will need to properly execute the technicalities of software publishing. These include the application of licences and copyright statements, understanding and assigning software authorship, crediting contributors, maintaining FAIR software metadata and publishing software artefacts. Finally, RSEs will need to understand the principles of software citation [11]. This concerns both the potential for reuse of their own work, which demands the provision of complete and correct up-to-date citation metadata for their software, as well as their own citation obligations deriving from building on previous work in the form of dependencies.

3.2.5 Using domain repositories/directories (DOMREP)

Almost all research software is developed within a specific scientific domain. Some software may be able to cross boundaries, but the majority will have a home domain, with which it needs to be able to interact. The RSE then needs to be aware of any domain specific repositories that will contain data sets, catalogues, and other domain specific artefacts, in addition to software. The RSE also needs to be aware of how their software can interact with the existing domain-specific data repositories. Finally, they need to be able to assess and use software repositories - domain-specific or generic - for publishing software with the relevant metadata.

3.3 Communication skills

RSEs do not work in isolation. They are embedded in a research group or work within a team of RSEs supporting particular research projects. RSEs often need to interact with and facilitate communication among colleagues, clients and contractors with a very broad spectrum of background-knowledge, specialisation, expectations, and experience whilst keeping diversity issues in mind. Communication skills are therefore crucially important. Team skills are also mentioned in common guides for SE such as the software engineering body of knowledge [2]. However, the interpersonal and organisational skills and the capacity for adaption required to work in a research setting warrants a much stronger emphasis on this field of competence.

3.3.1 Working in a team (TEAM)

Being able to work, and effectively communicate in teams is essential for RSEs. For example, RSEs need to be able to explain particular implementation choices made and may even need to defend them. Within a team of RSEs, code reviews improve knowledge transfer and increase team cohesion. The team might change on a project-to-project basis and might be comprised of colleagues with very different backgrounds including, for example, information technology (IT) staff, domain scientists and technicians working alongside software engineers. The shared values come into play and each RSE needs to ensure that these values are lived by and passed on to others. Senior RSEs may lead a team of RSEs.

3.3.2 Teaching (TEACH)

RSEs have many opportunities to teach. These range from inducting new colleagues to teaching digital skills either through short courses, for example from The Carpentries [12], or entire lecture series. RSEs may also act as mentors and consultants. Code review also includes aspects of the teaching skill.

3.3.3 Project management (PM)

The RSE should have knowledge of project management processes. At some institutes, project management tools and approaches differ between individual research groups, but it is useful if an RSE understands general structures of a PM scheme, or can bring in new ideas for improvement. Project management in research software engineering poses specific challenges (see USERS) that might require the capacity to flexibly adapt to changing conditions and deviate from common project management methods. Additionally, the RSE should know that SE offers various methods and approaches specifically tailored to management of software projects and products.

3.3.4 Interaction with users and other stakeholders (USERS)

Since research software is often developed as part of the research process itself, its requirements and specifications might change with the progression of research. Stakeholders of research software often change across different research projects or even within the course of one project. Roles in connection with research software are often in flux and diffuse. For example, a single person might be user, developer and project manager at the same time. Often this means it is necessary for an RSE to think "outside their comfort zone", whilst being able to convey their knowledge and experience to experts of other fields or persons at different hierarchy levels in a way they can understand more easily. These conditions pose specific challenges for requirements analysis, project management, training and support.

4 RSE specialisations

There is a large variety of RSE roles that embody a blend of the basic skills and competencies defined above. We now list some of these specialisations.

4.1 Specialisations within the core RSE competencies

Open science RSE Open science practices are increasingly required by research funding agencies. The Open Science RSE may accompany international teams sharing large code bases, ensuring FAIRness of the code (SRU, SP, research data management (RDM)) in accordance to their research plans and cycles (RC).

Project/community manager RSEs Large research software projects require somebody to maintain the overall perspective (PM) and to delegate tasks to project members (e.g. USERS, and TEAM). This role may or may not involve programming.

Teaching RSEs Teaching (TEACH) the next generation of scientists (RSEs or not) the core RSE skills while respecting the domain's culture requires being versed in didactics and pedagogy. Educators play a vital role in improving the quality of research software.

User interface/user experience designers for research software Good software is not only functional, but is also documented (DOCBB) and distributed (DIST, FAIR). It is also designed with users in mind (MOD).

4.2 Specialisations outside the core RSE competencies

${DOMAIN}-RSE Some RSEs may be particularly specialised in the intricacies of a research domain, (e.g. medical, humanities, physics).

Data-focused RSE Certain domains have a high demand for data science skills. This may be cleaning, sorting, analyzing and documenting data (RDM, data management plan (DMP)), perhaps anonymising or pseudonymising them (e.g. patient information in medicine). This role also includes transferring data sets securely and archiving them properly (FAIR).

Research infrastructure RSE Specialized scientific software is increasingly accessible via services residing on the Internet. Configuring these services also requires knowledge in administration, hardware, user management and access permissions (SysOps, IT, machine learning (ML)).

HPC-RSE HPC-focused RSEs help domain scientists to better use HPC resources. If working in a supercomputing centre, they may preconfigure generic software for the specific hardware (e.g. via compilation), providing ad-hoc batch scripts and instructions (including training courses) to run software efficiently. If evaluating access proposals to resources, they judge energy and efficiency aspects, which next to scientific relevance is of increasing importance. If working closer to users, they provide specialized knowledge for either manual or tool-assisted High-Performance Computing (HPC)-oriented refactoring, in collaborations focused on performance and porting.

ML-RSE Machine learning (ML) techniques stem from mathematical optimisation and statistics. Software for ML can be quite high level in its usage interface, and yet have HPC-grade internals. This motivates the need for an ML-RSE to not only know the given application domain (feature engineering) and the related software (be it for image data, linguistic data, etc) but also to chose the most adequate method combination (e.g. deep learning, reinforcement learning, neuro-symbolic learning, etc) Since ML techniques are inherently prone to biases leading to socially problematic consequences, this calls for additional caution, especially in training data.

Legacy RSEs While programming languages, libraries and frameworks may be evolving fast, research software tends to be nurtured and developed by generations of researchers, often without formalized code management practices (DIST). Experienced RSEs help modernising code, if necessary making it leaner and faster, often via work-intensive refactoring, and possibly exploiting code transformation tools. Notice the overlap with the HPC RSE.

Web-development RSE Web services have to offer users (which may include the public) a usable and accessible interface and yet be functional and interoperable, whilst also offering security and data integrity. The RSE balancing these contrasting needs communicates the trade-offs made with the involved stakeholders.

Legal-RSE Creating, composing, and then publishing research software having different sources and licenses has legal constraints. With the advent of data protection regulatory frameworks, and for compliance with export control regulations, the requirements are generally rising. This is where experienced RSEs may extend their knowledge into legal matters and interact more closely with lawyers and the local data protection officers.

5 Future work

This list and description of competencies is a first step to finding common ground around which to structure curricula, institutions, and teachers in this framework. An omission that we found and that we would like to highlight in order to spark a community discussion is that RSEs that choose explicitly a science-supporting role outside of research will not be eligible for funding under the statutes of many funding organisations that require at least a PhD.

To alleviate this and to give RSEs in leadership positions a means to become eligible for funding themselves, we see two possible parts of a solution. One is to allow for doctorates primarily based on software contributions to the scientific community. Secondly, we propose the introduction of new, standardised certificates and to officially accept them as PhD-equivalent concerning eligibility to be a Principal Investigator (PI). Beyond this discussion, a diverse set of publications on the topic RSE teaching is already in the making.

Within this set, we will work next on how to institutionalise education. In that publication, we will detail how we organise our institutions and what qualifications our teachers need to have in order to effectively communicate our values. We will put forward ideas on how to build up bachelor's and master's programmes, of which a glimpse can already be found in [4]. We will show how we intend to provide the necessary continuous education for RSEs after graduation, and we will connect that with the integration of RSEs into a mesh of community networks aimed at supporting research, while providing them with an inclusive social network that further facilitates lifelong learning. That publication will again intentionally be free of regional specifics, to also serve as a blueprint that other national RSE societies can build upon.

Online resources for courses are another important building block. Surveying and curating of existing resources is not carried out as a traditional publication, but it is made available as a continuously-evolving online database via the learn-and-teach project [13].

Finally, we plan to formulate a call to action, building on the previously mentioned publication on the necessary institutions, that spells out everything that is required to best support the continuous need for young RSEs to support digital science specifically in Germany.

6 Conclusion

This paper summarises a more extensive publication and the results of community efforts at RSE workshops in Germany, where people working in RSE related fields got together to figure out structures and ideas for educating newcomers to this field. One outcome of these diverse gatherings is that RSEs from differing fields gather around similar core concepts. At the same time, they share a vision of how to renew scientific research practice making extensive use of digital tools. In this work, we have tried to formalise these concepts. We have formulated a set of values that guide our actions in society, manifestly making RSEs part of the scientific community that shares the ideals of good scientific practice. At the same time, being close to software engineers, we cherish that we have to take responsibility for our tools. We listed core competencies that have been intentionally formulated abstractly without referencing any particular information-processing device. As expected, we have drawn equally upon notions from SE and other research fields, but found that we likewise require teamwork capabilities.

The listed values and competencies form a common denominator that unifies RSEs and enables them to identify with this role, in the knowledge that it is already or will soon become critically important for many areas of science. These competencies at the intersection of research and SE, coupled with a firm belief in team processes, make RSEs sought after on the job market and their values make them responsible members of a digital society. The result is a qualification profile which is highly attractive for young people.

At an institutional level, research-performing organisations have a growing interest in fostering RSE training to support the use of FAIR data and FAIR software in the academic world, a direction determined by new incentives created by scientific journals and librarians. How we update existing institutions and set up new ones that provide this education will be the topic of a follow-up paper.

7 teachingRSE biography

The teachingRSE project has been working on the associated full and more complete publication in [4] since 2023. The project was initiated by Heidi Seibold who came up with the original idea for the deRSE23 workshop in Paderborn. Heidi was joined by Jeremy Cohen, Florian Goth, Renato Alves, Jan Philipp Thiele, and Samantha Wittke to organise the initial deRSE23 workshop. Over the course of further workshops like the un-deRSE23 workshop in Jena, and the deRSE24 in Würzburg, the content of the paper was further refined and we are grateful to all particpants! In addition to the workshops the ideas were further developed during weekly meetings led by Florian Goth, together with the authors of [4], Renato Alves, Matthias Braun, Leyla Jael Castro, Gerasimos Chourdakis, Simon Christ, Jeremy Cohen, Stephan Druskat, Fredo Erxleben, Jean-Noël Grad, Magnus Hagdorn, Toby Hodges, Guido Juckeland, Dominic Kempf, Anna-Lena Lamprecht, Jan Linxweiler, Frank Löffler, Michele Martone, Moritz Schwarzmeier, Heidi Seibold, Jan Philipp Thiele, Harald von Waldow, and Samantha Wittke. The group gained more recognition, and the teachingRSE project is now an official working group of the de-RSE society and the special interest group on research software of the German Informatics Society (GI).

Acknowledgements

We thank all the contributors to the teachingRSE project!

We appreciate the comments and suggestions from Yves Vincent Grossmann, Wilhelm Hasselbring, and Bernhard Rumpe.

References

[1] Michelle Barker et al. “Introducing the FAIR Principles for research software”. In: *Scientific Data* 9.1 (2022-10), p. 622. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01710-x.

[2] Pierre Bourque, Richard E. Fairley, and IEEE Computer Society. *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0.* 3rd ed. Washington, DC, USA: IEEE Computer Society Press, 2014-01. 346 pp. ISBN: 978-0-7695-5166-1.

[3] Deutsche Forschungsgemeinschaft. *Guidelines for Safeguarding Good Research Practice.* Code of Conduct version 1.1. 2022-04. DOI: 10.5281/zenodo.6472827.

[4] F Goth et al. “Foundational Competencies and Responsibilities of a Research Software Engineer [version 1; peer review: awaiting peer review]”. In: *F1000Research* 13.1429 (2024). DOI: 10.12688/f1000research.157778.1.

[5] Florian Goth et al. *Foundational Competencies and Responsibilities of a Research Software Engineer.* arXiv. 2023-11. DOI: 10.48550/arXiv.2311.11457. arXiv: 2311.11457 [cs.SE].

[6] Don Gotterbarn, Keith Miller, and Simon Rogerson. “Software Engineering Code of Ethics is Approved”. In: *Communications of the ACM* 42.10 (1999-10), pp. 102–107. ISSN: 0001-0782. DOI: 10.1145/317665.317682.

[7] Simon Hettrick. *A not-so-brief history of Research Software Engineers.* Software Susitainability Institute. 2016-08. URL: https://www.software.ac.uk/blog/2016-08-17-not-so-brief-history-research-software-engineers-0 (visited on 2023-06-16).

[8] Paul Humphreys. *Extending Ourselves: Computational Science, Empiricism, and Scientific Method*. Oxford University Press, 2004-08-12. ISBN: 978-0-19-978596-4. DOI: 10.1093/0195158709.001.0001.

[9] Loïc Lannelongue et al. "GREENER principles for environmentally sustainable computational science". In: *Nature Computational Science* 3.6 (2023-06-26), pp. 514–521. ISSN: 2662-8457. DOI: 10.1038/s43588-023-00461-y.

[10] Katherine Richardson et al. "Earth beyond six of nine planetary boundaries". In: *Science Advances* 9.37 (2023), eadh2458. DOI: 10.1126/sciadv.adh2458.

[11] Arfon M. Smith et al. "Software Citation Principles". In: *PeerJ Computer Science* 2.e86 (2016). ISSN: 2376-5992. DOI: 10.7717/peerj-cs.86.

[12] The Carpentries. *The Carpentries*. URL: https://carpentries.org (visited on 2024-12-12).

[13] The teachingRSE project. *Learning and teaching RSE*. URL: https://de-rse.org/learn-and-teach/ (visited on 2024-12-12).