Teaching and Learning Research Software Engineering

Heidi Seibold Florian Goth Jan Linxweiler Jan Philipp Thiele
Jeremy Cohen Renato Alves Samatha Wittke Jean-Noël Grad
Fredo Erxleben

July 5, 2023

Contents

Working Title: Identifying foundational competencies of Research Software Engineers across all specializations.

Abstract: Being an outcome of a community workshop held in Paderborn, Germany in February 2023 this paper tries for the first time(FIXME: ? true?) to define which competencies are required to participate in modern digital sciences. Some of these competencies are required in more depth, therefore, giving rise to the trade of the RSE: scientific personnel that specializes in writing research software that facilitates work in all stages of the research cycle. Due to their generality, these competencies are often shared between RSE specializations, and we believe they are also relevant for domains outside of the RSE community.

But knowing a set of competencies is not enough, therefore we discuss explicitly how to make people aware that these skills are required and how these are taught(FIXME: Do we want to add this pedagogical dimension?). In order to also facilitate structural change in the German research institution landscape we will discuss the organizations and structures that support this change and educate new RSEs. The discussion in this paper is meant to be general. Therefore, we will discuss domain specific applications in an appendix.

Keywords: research software engineering, training, learning, competencies

Introduction

- background
  - RSEs have been around for a long time but the name is new.
  - RSEs have a specialist skill set that brings together technical and research knowledge.
  - Skills development traditionally provided largely through peer learning, self learning, introductory training courses not targeted specifically at RSEs, …
  - Requirements for RSE skills growing rapidly across all domains.
- past attempts, other initiatives
- contributions

Computers and software have played a key role in the research lifecycle for many decades. Traditionally, they were specialist tools used only in a small number of fields and the Computer Scientists who maintained and programmed them needed extensive technical training over several years to gain the necessary skills and expertise. Fast forward 50-60 years and software and computation are all around us, underpinning our everyday lives. This shift is also true within research.

With the ability to capture and process ever more data, undertake larger scale, higher resolution simulations and, increasingly, leverage new self-adapting approaches through Machine Learning, computers and software are now vital elements of the research process across almost all domains. However, this shift means that there is a much greater need for core research software skills across a vast array of research fields where these were not previously required.

The need to access both research data and software has been formalised with the FAIR principles: software and data need to be easily findable by both people and machines, and they also need to be accessible, interoperable and reusable. More recently the FAIR principles have been extended specifically to research software [3].

The people who focus on writing research software are now known as Research Software Engineers (RSEs) - a term that was coined a little over 10 years ago [30]. RSEs may work within one of the many Research Software Engineering teams that have been set up at universities and research organisations over the last decade, or they may be embedded within a research team. They may have a job title that officially recognises them as an RSE, or they may have a standard research or technical job title such as Research Assistant, Research Fellow or Software Engineer. Regardless of their job title, RSEs share a set of core skills that are required to write software, understand the research environment and ensure that they produce sustainable, maintainable code that supports reproducible research outputs. They are the ones who implement the FAIR principles that make digital research output more valuable. In order to do so the draw upon skills from traditional software engineering, established research culture and a commitment to being part of a team.

Developing and maintaining these skills is time consuming and often challenging. Part of the challenge is that there is not a standard pathway to becoming an RSE and, partly as a result of this, there is something of an ad hoc approach to training within the community. We also see increasing amounts of basic-level training materials that are great to put researchers or technical professionals on a path towards gaining significant RSE expertise, but the trail often ends as developing RSEs want to progress to intermediate and advanced level material. In particular, recent technology developments are ensuring that there is a growing need for specialist expertise, for example in areas such as making efficient use of high-performance computing infrastructure. This is an area where there is a skills shortage and a shortage of training materials.

In this paper, we look at the challenge of understanding the core competencies that underpin Research Software Engineering and the way that these competencies may be combined to help support a more coordinated approach to future RSE skills development. The paper builds on a workshop session held as part of the German Research Software Engineering Conference (deRSE23), held in Paderborn, Germany in February 2023.

[Information on key contributions to add]

[Overview of paper sections to add]

Intended Target Audience

While this paper is based on discussions held during a workshop at the deRSE23 conference we believe that the competencies formulated here have a far-reaching impact beyond the domain of RSE into adjacent fields of science. The most obvious users come from computer science, are HPC programmers with a background in physical sciences, or manage research data. Graduates from traditional STEM sciences with a focus on software or some library or IT staff will also find this paper interesting. However, these days most research involves some amount of data management, processing and visualisation and the role of RSEs is also becoming increasingly important in medical domains and the digital humanities. Access to central compute resources ranging from small departmental sizes to national facilities is becoming readily available. Additionally, pressure is growing from funding bodies to prioritise projects that generate archived, annotated, re-usable and potentially remotely executable data. These resources and requirements fall within the skill set of RSEs. They become a vital link to cross-pollinate computational skills and infrastructure know-how between domain scientists. Funders and research managers will find the discussion in this paper valuable in order to observe how software development in academia will be institutionalised. Finally, the strong emphasis on team-skills allows RSEs to be very employable in industrial workplaces.

Related Work and Activities

The challenges of understanding the current state of skills within the research software community and related areas, as well as identifying required competencies, developing training pathways and providing training materials are areas that are being looked at and addressed by various groups and projects. In this section, we highlight some of these other projects and activities.

Identifying skills and pathways

As an area that generally requires a range of advanced skills, High Performance Computing (HPC) is one field where there is ongoing work to identify relevant sets of skills for HPC practitioners and potential paths to develop these skills. The HPC Certification Forum has developed a "competence standard" (CS) for HPC that defines a range of skills and how they are related in the context of a skill tree [9], [34], [35]. This competence standard is currently being built upon by the CASTIEL 2 [16] project in collaboration with initiatives funded by the European High Performance Computing Joint Undertaking (EuroHPC JU) to create a framework for HPC certification [22]. Also looking at pathways and how different skills are related, the UNIVERSE-HPC project [42], funded under the UK's ExCALIBUR research programme [18], is looking to understand and develop training pathways to support the development of specialist skills in the HPC and exascale domains. The project is gathering open source training materials to develop curricula that support the training pathways that are underpinned by high-quality training materials.

- There are some projects / papers looking at skills pathways - if we're going to include a separate section on related work, as proposed here, this should probably be expanded to include more of this content?
- FIXME: include the ELIXIR part here

RSE-related Training Materials

A wide range of software-related training materials and supporting organisations exist within the research software community and beyond.

The Carpentries The Carpentries [1] is a non-profit entity that supports a range of open source training materials and international communities of volunteer instructors and helpers who run courses around these materials. The community also maintains the materials which are based around three core syllabuses - Software Carpentry, Data Carpentry and Library Carpentry. The training materials within these areas have been developed, reviewed and enhanced over several years ensuring that they represent best practice in training on these topics. The core Carpentries lessons are targeted primarily at the beginner level. However, the Carpentries Incubator [2] provides an environment for hosting additional community-developed training modules covering a wide range of other topics that have not gone through the peer review process of the core lessons. The material in the Incubator increasingly includes more intermediate-level training modules.

Coderefinery CodeRefinery [8] is a project currently funded by the Nordic e-Infrastructure and thus active primarily in the Nordics with the goal of teaching essential tools around research software development, that are usually skipped in academic education. CodeRefinery hosts a set of open source training materials including both beginner and intermediate level material and organizes multiple highly interactive large scale workshops per year. Skills learned from the workshops and/or materials allow researchers to produce more reproducible, open and efficient software and thus promote FAIR [43] research practices. One goal of the project is to evolve into a community project that seamlessly integrates with other initiatives. FIXME: elaborate on the integration part if it's relevant, else leave out.

Reprohack The ReproHack Team offers resources to host events where students and researchers can get together to try and reproduce the results of published papers with the methods described there or ideally with the software provided by the authors. FIXME: This is applied FAIR principles elaborate a bit on why this is related for RSE competencies? Or do we want to make the point, that this teaches reproducible science?

PRACE The Partnership for Advanced Computing in Europe (PRACE) [38] offers training in the form of massive open online courses (MOOCs), online and on-site training events at European HPC facilities (aggregated on various websites, e.g. EuroCC Training [17]), and white papers. While most training events are tailored for HPC-RSE, several recurring courses about programming languages (C++, Fortran, Python) are suitable for general RSEs, as they teach coding best practices, modern software design [31], project management and version control [24].

Helmholtz As part of its push towards a better RSE environment, the Helmholtz Association launched the Helmholtz Federated IT Services platform (HIFIS) [26] which provides educational material and trainings amongst other services for an audience of over 10.000 scientists in Germany and internationally. All of these materials focus on RSE basics to refresh and expand the software engineering knowledge for recent graduates or to update the existing knowledge in established researchers. They are published under OER licenses and can serve as either self-learning instructions or form the basis of a hands-on training. To allow these educational offers to be easier brought to the scientists, the Helmholtz Information and Data Science Academy (HIDA) [28] sustains a large network within the Helmholtz Association and beyond with a strong focus on graduate schools. Further RSE training offers within the Helmholtz context are provided by the Helmholtz-AI [25] and Helmholtz-Imaging [27] platforms as well as the Helmholtz Metadata Collaboration plaftorm [29].

Open Source Resources Due to the ever-evolving nature of skills and infrastructure in the RSE field, training material is often version-controlled, so that trainers can update it between iterations. For example, core lessons from the Carpentries and CodeRefinery are stored on GitHub, and any change is automatically mirrored to their website. Likewise, the reference work on RSE by Fogel [20] was released in its second edition as a living document [21].

Reference works are also available for self-study [20], [32].

The National Competence Center Sweden (ENCCS) [https://enccs.se/] provides instructor training material [14], [13] developed from Carpentries and CodeRefinery material, as well as lessons for HPC-oriented RSEs [15].

The (Intersect)[https://intersect-training.org/] project ….

(BSSW)[https://bssw.io/] …

SSI? [11] …

- FIXME: Add NFDI initiatives like edutrain.

Challenges

- Point out gaps
- What is missing
- domain application?

Depending on the specific domain there is a gap between the basic software carpentry courses and the required skills to build domain-specific research software. For example, scientists in the field of High Performance Computing (HPC) need to know how to make effective use of concurrency to speed up their simulations and communicate efficiently using message-passing interface (MPI) libraries. The same is true for researchers from other domains who make use of other specialized technologies, methods and/or tools. To bridge those gaps more specialized courses would be needed like the one mentioned in section Identifying skills and pathways for the HPC community.

Moreover, software development is a craft, i.e. it is not only about knowledge but also requires practical experience. Hence we need to create an environment that allows less experienced researchers to practice and gain experience efficiently. Ideally, this learning environment would allow less experienced scientists to be guided by more experienced RSEs. We know such practices e.g. from human medicine, where junior doctors first assist experienced doctors before they work independently. In the field of software development, this approach could be implemented in the form of peer programming, for example. The prerequisite for this, however, is that experienced academics get better career opportunities at German universities so that they do not leave for industry roles.

Results

Required Generic RSE skills

As it stands, the RSE role requires competencies in two fields. The "R", the person being a researcher, and the "SE" the software skills. And this hybrid nature is brought about, since RSEs need to apply their knowledge usually in teams. Therefore we structure our competencies among SE skills, research skills and team skills with key notions being the software and the research cycle and the scientific process. Since these skills are meant to be relevant in a broad setting and form the foundation for a specific specialization. We elaborate on some facets in tables.

Software Engineering Skills There are lots of software engineering curricula out there, that try to define which tasks a software engineer should be able to perform. A recent one highlighting some aspects in more detail than what we are doing here is [36].

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 softwares. An important part of reusability is that at least oneself, and ideally others, are able to understand what a piece of code aims to do and how to use the provided functionality, which is primarily achieved through a "clean" implementation and enhanced by documentation. This ranges from commenting code blocks to the usage of documentation (building) tools.

Building distributable libraries (LIBS) The RSE should be able to distribute their code with their domain/language specific distribution platforms. This almost always encompasses handling/documenting dependencies to other packages/libraries and sometimes requires knowledge of using build systems to enable interoperability with other systems.

Understanding the software lifecycle (SWLC) Software has a lifetime and this necessitates the respective strategies for its usage along the intended time scale.

Use repositories (SWREPOS) The RSE should be able to use public platforms to share the artifacts they have created and invite public scrutiny on them for public review.

Legal things (LEG) The RSE should know licenses and their respective domains for data or software. On an entry level, the competency is mostly about awareness. Namely that different (open source) licenses exist, that those might not be compatible with each other, and that use of third party software might restrict licensing of the resulting work.

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. Using that, an RSE should be able to make predictions about a software's behaviour. This is a required skill for common tasks like debugging, profiling, designing good tests, or predicting user interaction.

The research skills

Curiosity (NEW) RSEs gain their reputation from their effectiveness to interact with their domain peers. Therefore some curiosity together with a broad overview of the research field is required. A manifestation can also be the curiosity for new tools which is a great asset for an RSE. Lifelong learning then becomes not only bearable, but a motivation to work.

Understanding the research cycle (RC) Knowing that ones own research is not only a means to personal ends, but that one is part of a bigger cycle that involves a lot of other parties in and outside of your domain should foster an appreciation for the underlying principles of science like review and reproducibility.

Finding/discovering software and attribution (SD) One goal of FAIR software is to avoid unnecessary duplication of work by reusing existing work instead. To (re-) use software, individual researchers have to be able to find it and then to easily evaluate if the software actually suits their needs. Apart from functionality also licensing, integration with other software, expected sustainability and expandability have to be part of this evaluation. Finally, after obtaining/publishing results by modifying and/or using the software, the original authors need to receive proper attribution.

Use 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. Especially for data-driven research having software that is able to use existing sets and repositories is a valuable part. The RSE should be able to interact with the repositories of this specific domain.

Outside Party interaction (USERS) While in a traditional SE context, you might get away with not interacting with people outside your project. But in a research context, this will certainly be the case and involves users, other developers, upto funders. Additionally, this is oftentimes a two-way interaction with RSEs in a specific domain learning new findings, techniques, algorithms, etc. to be able to implement software that is up-to-date with the body of knowledge of that domain.

Team Skills

Teaching (TEACH) Working in a group means being able to effectively perform e.g. onboarding, or more formal teaching procedures to their colleagues. This includes tasks such as consulting and mentoring since these also often aim at educating people. We deliberately mention, that giving code reviews is also part of this skill, since Code review can be part of teaching people on improving their skills.

Project Management (PM) The RSE should have knowledge about project management. At some institutes, it follows the practices of the local research groups, but it is useful, if an RSE knows its place in a PM scheme, or can bring in new ideas for improvement.

Working in a team (TEAM) There are various facets to working in a team. They range from functioning in a team to leading a team. It includes following measures that increase team cohesion like performing code reviews.

Current Day Contextualization These skills, while already numerous are also on purpose generic. Concrete examples can be obtained by the outcome from the Paderborn workshop, where we asked learners and beginner RSEs of what they would liked to have learnt. Among the top five things mentioned were:

- Testing. This task is a manifestation of the SE competencies of DOCBB and MOD since a model of the software is required in order to write good test that facilitate understanding and documentation. Today this encompasses the knowledge of testing frameworks and CI/CD practices.
- Contributing to large projects. This is a topic that requires competency in SWREPOS , LEG, in order to understand the ramifications of sharing, DOCBB, since the contributed code has to be understood by others, and TEAM, since one is becoming part of the project depending on the involvement. Today this entails the effecive use of collaborative platforms like github/gitlab, honoring a projects Code of Conduct, and some knowledge of software licenses like the GPL.
- When or why to keep repositories private. This decision requires knowledge in the RC, to understand when it makes sense, USERS and TEAM in order to do accepted decisions and sometimes LEG. This requires domain and location knowledge in the sense that one should know what the practices of ones own institution are.
- Proper Development. This broad topic requires all of the SE skills. Of course these are the competencies that are the most fluid since they have to adapt at a high rate to the technological advancements. Additionally proper SE skills often require knowledge of TEAM, and PM. Today this means effective use of IDEs, static analysis tools, design patterns, documentation (for oneself and others), etc.
- Finding a community. This can be interpreted in two different facets. First we have the aspect of community building for a research project. Since this deals with software that is supposed to be used in research this requires knowledge of RC, USERS, and also NEW, in order to effectively interact with domain scientists. Today, an example is a presence on social media. The other TEAM-related aspect is the embedding of RSE graduates into the community of RSEs. We envision our RSE graduates to be a part in a strong network of other RSEs, tool-related communities and the classical domain communities. This point is further elaborated in How do we reach people in different stages of their careers

How much do different people need to know?

Now that we have the different competencies, we can explore various dimensions of these competencies, depending on their circumstances. A strong beneficiary of specialized RSEs can also be newly formed RSE centers at research institutions.

Career level At different career levels, differing skills are required. We have set this up according to the following separation often applied within a single project:

- Junior RSE: These are persons that have just started, but generally speaking they should have the skills to contribute to software projects
- Senior RSE: They have gained experience and can set the examples in the software project.
- Principal RSE: Their actual job description varies a lot. These may be RSE team leaders based in a professional services type role, or they may be professors or research group leaders based in a more academic-focused role. They are often the people responsible for bringing in the money that supports new projects and sustains existing projects. Generally speaking, they do not need to to be actively involved in the day-to-day technical tasks but they should be able to guide projects from both a technical and research perspective.

The required skills are distributed according to this table First Dimension: Career path e.g. Junior RSE -> Senior RSE -> PI scale (1->6) (less -> lot)

	Junior	Senior	Principal RSE(brings in funding)
DOCBB	should be able to write reusable building blocks	same as junior, but the quality should set the standard for the project, while following current best practices	should know the current best practices and point its people to the right resources.
LIBS	should be able to use package distribution platforms	same as junior, but should set the project standard and follow current best practices.	should ensure that their project is in an up-to-date distribution platform
MOD	should have a basic grasp of their piece of the software in order to use basic tools like a debugger	Should understand the characteristics of large parts of the codebase considering a variety of the metrics	Should understand the big idea of the software project in order to define the task that the software solves
SWLC	Awareness about the existence of the software lifecycle.	Should know which decisions lead to technical debt.	Should know in which part of the lifecycle their project is and how to steer development/project resources accordingly.
SWREPOS	Seamless interaction with the swrepo of their project is a must	Should be well-versed in the intricacies of a swrepo, and probably interact with multiple projects' repo's	Should be able to effectively interact with swrepos and especially the interaction with the connecting projects.
LEG	Awareness about legal intricacies about sharing code	Should be able to give advice on legal issues and resolve the most common issues	same as Senior RSE
NEW	Some curiosity required to fit into research teams	same as junior, but a curiosity to enhance the code base is required	Curiosity to know in which direction to steer the project is required

	Junior	Senior	Principal RSE(brings in funding)
RC	Awareness about the RC	should know the position of the project in the RC	Should know what is necessary for the project to fit into its position in the RC
SD	Should be aware about tools for SD	Should be able to find sth. with SD tools	Should be able to effectively find sth. with SD tools and be able to evaluate and perform the integration of a library into the project.
DOMREP	The RSE should be able to interact with the domain repository	same as junior RSE	same as junior, and should know about how it fits into workflows surrounding these domain repositories
USERS	The RSE should be able to communicate with non-SE users of the project	same as junior	same as junior, and take feedback into account of the steering
TEACH	should be able to perform simple peer-to-peer onboarding tasks	should be able to explain logical components to other RSEs	Should be able to effectively communicate about all large-scale parts of the project.
PM	Awareness about the employed project managemement method	Should be able to use the employed PM method	Should be able to design and adapt the employed PM method.
TEAM	Should be able to work in the team in order to effectively fulfill the given tasks. Should be able to learn from code review.	Should be able to break down tasks into more easily digestable sub-tasks	Should be able to lead the team and set the respective direction.

Academic Progression / Career Path? (Help me for better title) Modern digital science requires some digital proficiency at every level. To be a bit more precise, these are how we define the academic levels:

- Bachelor: These are people in their undergrad studies, that mostly consume science/knowledge. They should also learn about the existence of certain digital structures.
- Master: Ultimately, their study should have brought them to a level, where they can participate in science, hence they should be able to use “some” digital structures.
- PhD: Under guidance they perform independent research and hence they should get to know all relevant structures.
- PostDoc: Independent researchers, they are proficient users of all tools.
- PI/Professor: Experts in their field, they should be able to give proper guidance to their students on which digital tools are currently relevant.

It is important to note that the following table does not reflect the current state of academic training and research institutions. Instead, it summarizes the discussions with and between workshop participants at different levels of academic progression on what they would have liked to learn at an earlier stage or know before starting their current position. While individuals already work at implementing some of these changes and teaching these skills it has not yet reached a systemic level.

Additionally, this table tries to cover all domains that rely on software tools in at least a basic level. Certain fields, e.g. sciences relying on simulations, might require higher skill levels in the SE competencies as software development is a large part of their actual research.

	Bachelor	Master	PhD	PostDoc	PI/Professor
DOCBB	They should be aware that RSEs exist and that software has different quality aspects	Same as Bachelor	They should know where they can get help, and maybe able to use libraries	same as PhD	They should know the skills of an RSE and when they might need one in their group
LIBS	They should be aware that RSEs exist and that there are tools available in their domain	They should be aware that there are tools that they can use in their research and maybe are able to use these libraries	same as Master, but able to use libraries	same as PhD	They should be aware of the output of RSEs and motivate their students to use developed tools
MOD	It is sufficient to consider digital tools as black boxes	It is sufficient to be able to *use* software as black boxes	same as Master, but being able to write bug reports	same as PhD	same as PostDoc
SWLC	Awareness of the SWLC	Know that one depends on software in their own research	Being able to evaluate software for their research	same as PhD	Should be able to judge the sustainability of the performed research

	Bachelor	Master	PhD	PostDoc	PI/Professor
SWREPOS	Should know that swrepos exist	same as Bachelor	same as Master, but should be able to find information on them	same as PhD	same as PostDoc, but should be able to follow the interactions among different projects relevant for their research
LEG	Should know that mixing/using software has legal issues and whom to ask	same as Bachelor	same as Bachelor	same as PhD, but should know some simple Open Source guidelines	same as PostDoc, but should know the relevant patterns for their domain and sensitive their students
NEW	Still consumers of lectures	same as Bachelor	Curiosity for their research is required, curiosity for digital tools helpful	same as PhD	same as PostDoc and expert in their field
RC	An awareness that research follows a cycle	Know that research follows a cycle and locate their masters thesis' stages in it.	Same as Master, but applied to the PhD. Additionally awareness about interaction with services	Same as PhD. But proficient in the domain	Same as PostDoc, but ability to lead a topic

	Bachelor	Master	PhD	PostDoc	PI/Professor
SD	They should know that their domain has relevant tools	same as Bachelor	Should know how to find full applications for their research	same as PhD	Should motivate their students to reuse existing tools
DOMREP	Should be aware that their domain has repos	same as Bachelor	Should be able to interact with their domain repos	Proficient users of their domain repos	same as PostDoc
USERS	Should be aware that they are users of a software	same as Bachelor	Should be aware that their user view is different from the developer, in order to write bug reports	same as PhD	Should be able to contribute meaningfully to the steering decisions of the software in their fields
TEACH	Ability to peer-to-peer teaching	Small exercise groups	Ability to supervise a student.	Ability to supervise students and create a course?	Ability to guide students. Give full-size lectures
PM	Awareness about project management optional	Awareness that research teams are structured according to some project management	same as Master, or more depending on structure of research	same as PhD	Should know about the required project management they require for their group
TEAM	Awareness that research is often performed in groups	Ability to work in their group for doing their master's thesis	same as master	same as master	Should be able to lead a research team

Project team Size Some explanation of the team sizes:

- individual: A single person working on their research software
- Small team(~4 persons) This is a small team, that has decided to work together on something
- Organizations(>10 persons): These are big organizations with clear structures and a bigger degree of specialization.

	individual	small team	organization
DOCBB	you might get away with less satisfactory code, as long as the product is OK	think about your colleagues	your organization most likely has guides here
LIBS	you will only be successful if your artifact is usable by others	same here	your organization probably has rules here
MOD	you should precisely know what your entire code is doing where	you should know what your part is doing and have a feeling about the others contributions	You should know what your small part is doing
SWLC	it's you and your software	You should know the Bus factor	The organization takes care of that
SWREPOS	you need academic credit.	same here	your organization probably has rules here
LEG	you carry the responsibility	someone in your group needs to take car of this	your organization will have specialized people for it
NEW	You need a motivation to do this alone	?	Not so much, since other people might do this task
RC	?	you should maybe talk among your peers where your software fits in	You define your research cycle
SD	you need to be able to build on other work to be successful	same here	there might be someone in your organization who does this
DOMREP	You're doing science in a domain	there should be a person in your team who knows how to do it	your organization might have specialists for that, but some basic familiarity
USERS	at one point you hope to have users	same here	maybe you have specialists for outreach
TEACH	N/A	able to peer teach	teaching to groups

	individual	small team	organization
PM	Not much required	able to follow checklist	Working with PM tools, or use them for organization
TEAM	N/A	should be able to give equal feedback to their colleagues	should be able to work within their role

Bonus points may be distributed if managing teams remotely

BIO Excel framework

RSE specializations

What we have defined above are intended to be base skills that an RSE irrespective of domain, place, and time should know about. But not all RSEs are created equal, they specialize in different areas, some of which we want to present below. Many of the specializations may overlap, so the same RSE might for example work on data management and on Open Science.

HPC-RSE RSEs with a focus on High Performance Computing (HPC) have specialist knowledge about programming models that can be used to efficiently undertake large-scale computations on parallel computing clusters. They may have knowledge of (automatic) code optimization tools and methods and will understand how to write code that is optimized for different types of computing platforms, leveraging various efficiency related features of the target hardware. They are familiar with HPC-specific package managers and can build dependencies from sources. They also understand the process of interacting with job scheduling systems that are often used on HPC clusters to manage the queuing and running of computational tasks. HPC-focused RSEs may be involved with managing HPC infrastructure at the hardware or software level (or both) and understand how to calculate the environmental impact of large-scale computations. Their knowledge of how to run HPC jobs and write successful HPC access proposals can be vitally important to researchers wanting to make use of HPC infrastructure.

They may also be familiar with High-Throughput Computing (HTC) and manage a network of heterogeneous compute resources, typically desktop workstations equipped with multicore processors and possibly GPU accelerators. They can apply their node-level performance engineering skills to maximize utilization of the available resources. Finally, they typically have expert knowledge in at least one compiled language, and can assist domain scientists who have excellent command of scripting languages but only a cursory understanding of compiled languages get up to speed with compiled software.

Research Infrastructure RSE This RSE is interested in SysOps and sets up infrastructures for and with researchers. This RSE, therefore, requires a deep knowledge of physical computer and network hardware. FIXME: While required, is this an RSE? this sets off the usual infrastructure vs. research discussion....

Web-Development RSE This RSE is skilled in web applications, front- and/or backend, and/or building and using APIs, for example for research data portals or big research projects. Ideally, this RSE should also have knowledge about (web) accessibility to allow a broad range of researchers or even the public to use the resulting applications. Therefore a deep knowledge of web skills is a required skill for this RSE.

Legal-RSE With the prevalence of software, we foresee the need for RSEs that specialize in legal questions around software. They are the go-to person if people have a question about licensing, mixing and matching software, and/or patenting.

Data-focused RSE RSEs working at the flourishing intersection between data science and RSE. They are skilled in cleaning data and/or running data analyses and can help researchers in setting up their analysis pipeline and/or research data management (RDM) solutions. When the field requires research on sensitive data or information, e.g. patient information in medicine, this RSE should have knowledge about secure transfer methods and/or ways to anonymize the data.

OpenScience RSE Open Science and FAIRness of Data and Software are increasingly important topics in research, as exemplified by the demand of an increasing amount of research funding agencies requiring openness. Open Science RSEs can help researchers navigate the technical questions that come up when practicing Open Science, such as “How do I make my code presentable?”, “What do I need to consider when it comes to licensing?”, or “How can I use version control / automation for my project?”.

Project/Community manager RSEs When research software projects become larger, they need someone who manages processes and people. Building a community around a research project is an important building block in building sustainable software, so these RSEs play an important role, even if they do not necessarily touch much of the code themselves.

Teaching RSEs RSEs who focus on teaching the next generation of researchers and/or RSEs play a vital role in quality research software.

${DOMAIN}-RSE While software is the lingua franca of all RSEs there will be RSEs that have specialized in the initricacies of one particular research domain, such as medical RSEs, digital humanities RSEs or physics RSEs.

Optional RSE competencies -> Maintenance RSEs

Oftentimes, a significant amount of effort in (research) software development needs to be spent on maintenance to ensure that software remains useful for researchers now and in the future. The research environment is constantly changing and this can also apply to the software requirements. Accordingly, software often needs to be adapted continuously. If it isn't, the software can reach a point where it simply isn't useful to the researchers anymore. To avoid this, regular work needs to be invested. While ensuring maintenance and sustainability of research software is of huge importance to the communities that build and use it, a particular challenge is that it's often very difficult to obtain ongoing research funding for software maintenance tasks. As a result, when a project that developed or extended a piece of software finishes, it can often be the case that support for the software fades as team members move on to other research, academic or RSE roles, or become busy with other funded work. While this is not a core concern of this paper, we wanted to highlight this important issue that is frequently faced when working with software in the research community. With regard to which additional competency is required, these are people having experience with ancient software stacks that are not anymore part of the general curricula(think of COBOL and FORTRAN).

FIXME: I think it would be nice if we could move each of these optional competencies to a different specialization.

How do we reach people in different stages of their careers?

Many current RSEs have found their way to being an RSE during their doctoral studies. This transition usually happens slowly. You start programming a tool, and someone else likes it, it becomes known that you have programming skills and suddenly you are the RSE of the group that everyone would like to have in their projects. If you enjoy this role, you need to be aware that there is a RSE career path as well as that specialized training materials exist. One place to generate awareness of the career option and training is universities' doctoral onboarding processes or right thereafter. RSE training could be offered as elective courses at universities organized by some central organization. RSE could be presented as a career path in suitable events. Since many RSE-minded people also at some point find their way to an HPC cluster, mailing lists of said clusters could be utilized to advertise RSE courses. One important aspect to think about is also the wording in the advertisement. Potential future RSEs might not know the term yet or know that the course advertised includes topics that are of interest to them. If the university or organization has a GitHub/Lab organization/project, having a banner there might reach the right people. Most important is that people working in IT helpdesks know about the courses offered so that they can advise students/researchers on visiting the course/reviewing the materials if related questions are asked. For an RSE it is important to be a part of a network with other RSEs, irrespective of the career level. A perfect first step for forming this network is topic-related conferences, workshops or meetups. Beyond that, there is the broader RSE community organized at the local and regional level with chapters or local/regional communities, at the national level with societies and the international RSE society. Each of them offers possibilities for connecting within or beyond an individual institution and is a great way to find like-minded people to grow a wider network and thereby facilitate the sharing of information on interesting events or help each other out. This available layered network can greatly benefit the RSE in finding help with issues outside of their own comfort zone and provides a welcoming, social safety net providing a home for the RSE. Since we feel providing aspiring RSEs this net is of utmost importance we envision compulsory events introducing that to young RSEs. Qualification badges are another venue, that RSEs to find people with the same technical interest.

Short primers on RSE skills, infrastructure and good coding practices can be found in field-specific scientific articles and conference proceedings, such as [40], [4], [39], [37], [44], [41], [12], [10], [19], [23], some of which are specifically tailored to group leaders, institutions and scientific journals rather than RSEs [7], [6], [33]. Scientific journals have the advantage of reaching a large spectrum of research scientists at all stages of their career.

Organizational Infrastructures

So we have defined our set of competencies that we feel every RSE should possess. Table 2 above nevertheless already hints at the fact that some RSE skills are required during the domain studies, while Table 1 tells us that we also need an ongoing qualification programme for people that want to become specialized RSEs. In order to set up a proper educational scheme we need to discuss two more items:

- Who are our teachers?
- How is this teaching organised?

The teachers

What issues are trainers facing today? There are already some people out there who are teaching RSE related topics sometimes in university structures, but often outside of formal structures. The community discussion shed some light on the issues our trainers are facing now. Currently, they are often teaching workshop like formats in research institutions.

- There are outreach issues. We emphasize that there are two dimensions to this: First it is important that we inform students that workshops exist, and then, the more important part, we also need to motivate people to invest the time for a workshop. [5]
- Adaptation of material to the target audience has been identified as a time consuming task.
- Organization and preparation is a challenge, since currently no standardized formats exist.
- Expectation management of students. Existing knowledge of students is often diverse.
- Language barriers. This can range from the use of technical jargon up to the disparities of you teaching in a foreign language.
- Setting up a feedback loop that facilitates a reflection of the workshop for the teacher.
- staying up-to-date with fast-moving RSE topics.

What mindset makes up a good teacher Irrespective of where people come from they need to have the proper mindset to properly foster aspiring RSEs.

- "If you want to go fast, go alone; if you want to go far, go together?"
- Not every "good" scientist wants to become a "good" software engineer, too!

Where do we get our teachers from The community discussion brought about the need for a mixture of people, thereby the education of aspiring RSEs will involve people from close domain sciences or experienced RSEs and people that have respective additional skills to teach RSE competencies to the new generation. In that respect, this follows the carpentries model that offers certifications but is still open to non-certified trainers. We highlight and emphasize, that since a topic like RSE education, is constantly evolving, trainers strongly require the opportunity to and the recognition to educate themselves. Therefore our teachers will be sourced from the workplace but there will also be certified RSE Trainers. (FIXME: in classical university speak, these would be people who have done their habilitation on that topic, right?)

We propose to create common Infrastructures that can be utilized in this ongoing effort to professionalize the RSE education further, to easily share education resources across the country. (FIXME: DETAIL ME FURTHER!!!!)

Organization of teaching

Certificates With the ever-growing demand for RSEs in science it could be helpful for people to earn certificates for skills needed in certain RSE positions. This would possibly make hiring easier and could incentivise researcher to go through proper courses on these skills instead of learning on the go. For certain skills it would also be good for finding jobs outside academia, e.g. in industry where certain practices are already state-of-the-art. However, these certificates are only helpful if there is a certain level of standardization, which would require a central authority or collaboration between multiple stakeholders to decide on contents and allow participating institutions to issue these certificates. Additionally, it can be excluding capable applicants that already use these skills but never got a certificate for it.

The possible types of certificates can of course differ. Badges are increasingly popular, from personal badges rewarding contributions to a specific community, e.g. Fedora Badges or Github Achievements, or project badges highlighting coding practices, e.g. Build and CI status or code coverage. RSEs are very likely subject to life-long learning and personal, technical badges are one possibility for older RSEs to showcase that they posess a certain technical skill. Classical attendance sheets for courses are another option. To further incentivise participation by students, some of these courses might also award academic credit points like ECTS and benefit them on their way to graduation.

Having certificates provides finally a clear understanding of which tasks an RSE can perform and thereby helps defining the career path and the job description. A big demand for specialized RSEs will certainly come from the newly established RSE centers at research institutions that require skilled people to fill their vacant positions. And using the certificates, the demand can now be satisfied with people offering this skill.

Some exemplary skills for which courses are already held are version control tools like git, HPC topics like multithreading, MPI and GPU computations, FAIR principles.

A possible graduation path within the classical university structures We have put forward the idea that familiarity with research is a prerequisite for an RSE in order to be able to work effectively in the research space and in collaboration with researchers. In this particular example, we consider a path into RSE via a traditional university route involving Bachelors and Masters degree studies that include an RSE element. However, we recognise that there are other routes into an RSE career and these are increasing. For example, some RSEs come from an industry background, others may come through apprenticeship or similar programmes. In both cases, gaining knowledge of the research lifecycle and understanding the ways that researchers work towards solutions to research challenges is something that can be developed on-the-job alongside training opportunities and the chance to work directly with researchers. This leads naturally to the question, whether it is possible to become an RSE without a home research domain. With software being a core element of the research process in so many different domains, it is not helpful for everybody working in RSE to have a background in computing or software engineering. Indeed, we consider it much more useful if new graduates looking to work in the RSE space come from a wide range of different domains. Expertise, beyond software development skills, in another research domain can be an important element of an RSE team being able to support RSE projects in that domain. Assuming for a moment, that people have done their masters studies in a particular domain, e.g. from the natural sciences, and that we can assume that the lectures to that point contain a mixture of domain specific content and RSE specific content (A good starting point for an RSE in the natural sciences), then we come to the question of the topic of the masters thesis. In order for young RSEs to get their research experience we believe it is necessary that already in their master's thesis young RSE students are given computational research tasks that can be solved with the RSE specific skills of that domain. This gives them a Master's degree of a ${DOMAIN}-RSE that has learnt in their lectures a research domain specific part and a software engineering specific part, and enabled them to get a first dip into actual science in their master's thesis. Of course, the next question for their future is whether a master's degree enables them to really be effective parts of a research group. While we accept this is something of a generalisation, we argue that this is most likely not the case since undertaking a PhD provides a much more extensive set of research training and experience that can be vital for a researcher's future. Research environments different internationally but in many cases there are formal barriers in the research landscape that require a PhD (e.g. eligibility for funding). Hence a PhD is required to actively participate in science and why we argue the regular RSE should do a PhD that on one end combines knowledge of a research domain with software engineering heavy task such that both pillars are suitably covered and they were able to observe how research functions.

Specialised Master's Programs On the other hand, when pursuing a PhD, scientists are increasingly required to do RSE-type work as part of their research as data and computation are becoming part of research tasks in a huge range of fields. It is not uncommon for researchers to be faced with RSE topics for the first time, because it has not been part of their academic curricula. Many are faced with a steep learning curve that requires them to invest a huge amount of time to catch up. Naturally, many would only invest as much as necessary to get the job done regardless of whether the solution is sustainable or not. Support from RSEs is one way to resolve this challenge. Another would be to lay more effective foundations for future RSE work at a much earlier stage in undergraduate/postgraduate studies. We see scope for establishing dedicated RSE Master's programmes which specialise in developing RSE skills and practices. Some universities already offer dedicated master's programs in some domains. Examples would be Computational Sciences in Engineering (CSE) or Bioinformatics. Where appropriate similar programs should also be established in other domains.

Required Next steps

Implementation Strategies

- Ideally over time scientific software engineering becomes part of the curricula at universities.

Academic Considerations

- Awareness of existing teaching programs
- "Branded" Add-on courses
- External Institutions provide resources
- fully recognized in the academic system. Students get ECTS points.
- Bachelor/Master specializations

Broader Considerations

- Instilling more respect for people that want to educate themselves for digital competencies
- Outreach to people that now have the feeling that they require this training.

Conclusion

We have indentified the RSE has an individual that contributes to research teams with their knowledge about digital tools. Then we have defined generic core competencies from the pillars of Software Engineering, Research and Team processes. We fleshed them out with some possible specializations of RSEs. Given the competencies and a demand(FIXME: Do the calculation) in the research landscape for them we moved on to define who the teachers are for this new field. We closed with a discussion of possible structures and organization forms that educate new generations of RSEs in more structured programs than what is available today(FIXME: this is currently aspirational). Therefore this closes the gap, that the research landscape requires RSEs, but there are no structures where these persons are educated, by detailing the career path that a young person might want to take to become an RSE.(FIXME: also aspirational...)

Appendix

An applied example curriculum

An example of a possible career path

- We can follow Kim, who has been the protagonist of the original deRSE Paper.

References

[1] The Carpentries. URL: https://carpentries.org (visited on 06/16/2023).

[2] The Carpentries. URL: https://carpentries-incubator.org/ (visited on 06/16/2023).

[3] Michelle Barker et al. "Introducing the FAIR Principles for research software". In: *Scientific Data* 9.1 (Oct. 2022), p. 622. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01710-x. URL: https://doi.org/10.1038/s41597-022-01710-x.

[4] Susan M. Baxter et al. "Scientific Software Development Is Not an Oxymoron". In: *PLoS Computational Biology* 2.9 (2006), e87. DOI: 10.1371/journal.pcbi.0020087.

[5] CASTIEL and EuroCC Network. "Best Practice Guide: How to Find New Attendees for Training Courses". In: *Training Best Practice Seminar (CASTIEL)* (Jan. 20, 2022). EuroCC, Jan. 2022. URL: https://www.eurocc-access.eu/wp-content/uploads/2022/04/20220401_Best_Practice_Guide-Training_Best_Practice_Seminar_final.pdf.

[6] Neil Chue Hong. “Minimal information for reusable scientific software”. In: *2^{nd} Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2)* (New Orleans, LA, USA, Nov. 16, 2014). figshare, July 2014. DOI: 10.6084/m9.figshare.1112528.

[7] Neil Chue Hong, Brian Hole, and Samuel Moore. “Software Papers: improving the reusability and sustainability of scientific software”. In: *1^{st} Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE1)* (Denver, CO, USA, Nov. 17, 2013). figshare, Sept. 2013. DOI: 10.6084/m9.figshare.795303.

[8] *Coderefinery.* CodeRefinery. URL: https://coderefinery.org (visited on 06/16/2023).

[9] *Competencies.* The HPC Certification Forum. URL: https://www.hpc-certification.org/cs/ (visited on 06/16/2023).

[10] Tom Crick, Benjamin A. Hall, and Samin Ishtiaq. “Reproducibility in Research: Systems, Infrastructure, Culture”. In: *Journal of Open Research Software* 5.1 (Nov. 2017), p. 32. DOI: 10.5334/jors.73.

[11] Stephen Crouch et al. “The Software Sustainability Institute: Changing Research Software Attitudes and Practices”. In: *Computing in Science & Engineering* 15.6 (Nov. 2013), pp. 74–80. ISSN: 1558-366X. DOI: 10.1109/mcse.2013.133.

[12] Michael R. Crusoe and C. Titus Brown. “Walking the Talk: Adopting and Adapting Sustainable Scientific Software Development processes in a Small Biology Lab”. In: *Journal of Open Research Software* 4.1 (Nov. 2016), e44. DOI: 10.5334/jors.35.

[13] ENCCS. *ENCCS Instructor Training.* Training Material to Train the Trainer. EuroCC National Competence Center Sweden, Nov. 2022. URL: https://www.eurocc-access.eu/wp-content/uploads/2022/12/EuroCC_NCC_Sweden_instructor_training_workshop.pdf.

[14] *ENCCS Instructor Training.* ENCCS. URL: https://enccs.github.io/instructor-training/ (visited on 06/16/2023).

[15] *ENCCS lessons.* ENCCS. URL: https://enccs.github.io/instructor-training/enccs-lessons/ (visited on 06/16/2023).

[16] *EuroCC 2 and CASTIEL 2: Promoting HPC to boost digital skills, jobs and industrial competitiveness in Europe.* EuroHPC Joint Undertaking. URL: https://eurohpc-ju.europa.eu/eurocc-2-and-castiel-2-promoting-hpc-boost-digital-skills-jobs-and-industrial-competitiveness-europe-2023-02-03_en (visited on 06/16/2023).

[17] *EuroCC Training.* EuroCC. URL: https://www.eurocc-access.eu/services/training/ (visited on 06/16/2023).

[18] *ExCALIBUR - Exascale Computing ALgorithms & Infrastructures Benefiting UK Research.* ExCALIBUR. URL: https://excalibur.ac.uk (visited on 06/16/2023).

[19] J. Fehr et al. “Sustainable Research Software Hand-Over”. In: *Journal of Open Research Software* 9.1 (Apr. 2021), p. 5. DOI: 10.5334/jors.307.

[20] Karl Fogel. *Producing Open Source Software. How to Run a Successful Free Software Project.* 1st ed. O’Reilly Media, Oct. 2005. ISBN: 978-0-596-00759-1. URL: https://www.oreilly.com/library/view/producing-open-source/0596007590/.

[21] Karl Fogel. *Producing Open Source Software. How to Run a Successful Free Software Project.* 2nd ed. https://www.producingoss.com, Jan. 2017.

[22] Governing Board of the EuroHPC Joint Undertaking. *Amending the Joint Undertaking’s Work Programme and Budget for the year 2023 (Work Programme and Budget Amendment no. 1).* EuroHPC JU Decision No 03/2023. EuroHPC Joint Undertaking, Mar. 2023. URL: https://eurohpc-ju.europa.eu/system/files/2023-03/Decision%203.2023.-%201st%20Amendment%20WP%202023.pdf.

[23] Alan Grossfield. “How to be a Good Member of a Scientific Software Community [Article v1.0]”. In: *Living Journal of Computational Molecular Science* 3.1 (Jan. 2022), p. 1473. DOI: 10.33011/livecoms.3.1.1473.

[24] C. Guillen and C. Navarrete. *Introduction to C++.* Leibniz Supercomputing Centre. URL: https://doku.lrz.de/2022-09-21-introduction-to-c++-hcpb2s22-11497182.html (visited on 06/16/2023).

[25] *Helmholtz AI.* Helmholtz. URL: https://www.helmholtz.ai/ (visited on 06/16/2023).

[26] *Helmholtz Federated IT Services (HIFIS).* Helmholtz. URL: https://hifis.net (visited on 06/16/2023).

[27] *Helmholtz Imaging.* Helmholtz. URL: https://helmholtz-imaging.de/ (visited on 06/16/2023).

[28] *Helmholtz Information & Data Science Academy (HIDA).* Helmholtz. URL: https://www.helmholtz-hida.de (visited on 06/16/2023).

[29] *Helmholtz Metadata Collaboration (HMC).* Helmholtz. URL: https://helmholtz-metadaten.de (visited on 06/16/2023).

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

[31] Klaus Iglberger. *Modern C++ Software Design.* Leibniz Supercomputing Centre. URL: https://doku.lrz.de/2022-10-26-modern-c++-software-design-hcpa1w22-11497188.html (visited on 06/16/2023).

[32] Damien Irving et al. *Research Software Engineering with Python. Building software that makes research possible.* CRC Press, 2021. ISBN: 978-1-003-14348-2. DOI: 10.1201/9781003143482.

[33] Matthias Katerbow and Georg Feulner. *Recommendations on the Development, Use and Provision of Research Software.* Zenodo. 2018. DOI: 10.5281/zenodo.1172988.

[34] Julian Kunkel et al. "The HPC Certification Forum: Toward a Globally Acknowledged HPC Certification". In: *Computing in Science & Engineering* 22.4 (2020), pp. 110–114. ISSN: 1521-9615. DOI: 10.1109/MCSE.2020.2996073.

[35] Julian Martin Kunkel et al. "One Year HPC Certification Forum in Retrospective". In: *The Journal of Computational Science Education* 11.1 (Jan. 2020), pp. 29–35. ISSN: 2153-4136. DOI: 10.22369/issn.2153-4136/11/1/6.

[36] Carl Landwehr et al. "Software Systems Engineering programmes a capability approach". In: *Journal of Systems and Software* 125 (2017), pp. 354–364. ISSN: 0164-1212. DOI: 10.1016/j.jss.2016.12.016.

[37] Felipe da Veiga Leprevost et al. "On best practices in the development of bioinformatics software". In: *Frontiers in Genetics* 5 (July 2014). DOI: 10.3389/fgene.2014.00199.

[38] *Partnership for Advanced Computing in Europe.* PRACE. URL: https://prace-ri.eu/ (visited on 06/16/2023).

[39] Andreas Prlić and James B. Procter. "Ten Simple Rules for the Open Development of Scientific Software". In: *PLoS Computational Biology* 8.12 (Dec. 2012), e1002802. DOI: 10.1371/journal.pcbi.1002802.

[40] K. V. Roberts. "The publication of scientific Fortran programs". In: *Computer Physics Communications* 1.1 (July 1969), pp. 1–9. DOI: 10.1016/0010-4655(69)90011-3.

[41] Victoria Stodden and Sheila Miguez. "Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research". In: *Journal of Open Research Software* 2.1 (July 2014), e21. DOI: 10.5334/jors.ay.

[42] *UNIVERSE-HPC: Understanding and Nurturing an Integrated Vision for Education in RSE and HPC.* UNIVERSE HPC. URL: https://www.universe-hpc.ac.uk (visited on 06/16/2023).

[43] Mark D. Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific Data* 3.1 (Mar. 2016), p. 160018. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18. URL: https://doi.org/10.1038/sdata.2016.18.

[44] Greg Wilson et al. "Best Practices for Scientific Computing". In: *PLoS Biology* 12.1 (Jan. 2014). Ed. by Jonathan A. Eisen, e1001745. DOI: 10.1371/journal.pbio.1001745.