The Research Software Encyclopedia: a community framework to define research software

Journal of
open research software

The Research Software Encyclopedia: A Community Framework to Define Research Software

SOFTWARE METAPAPER

VANESSA SOCHAT
NICHOLAS MAY
IAN COSDEN
CARLOS MARTINEZ-ORTIZ
SADIE BARTHOLOMEW

*Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

The Research Software Encyclopedia is a community driven, open source strategy to define the term "research software" in different contexts. It consists of several elements: a base library to manage a database of software, criteria and taxonomy items that can be used to answer questions about the software in the database, and several ways for an interested party to interact. A community database is stored in version control (GitHub), and by way of providing and updating this database, the Research Software Encyclopedia takes a strategy of small contributions over time to grow a valuable resource. Using a community-driven open source approach offers a number of advantages over attempting to derive a single, holistic definition for research software. First, it takes into account the context under which the definition is considered. Second, community and scoped contributions to specific components of the task are easy. Third, it provides a resource that can be extended to other use cases. Finally, this initiative creates a solution that requires no grants or other funding to maintain, increasing its ability to grow, adapt, and evolve over time.

CORRESPONDING AUTHOR:
Vanessa Sochat
Computer Scientist, Lawrence Livermore National Lab, US
sochat1@llnl.gov

KEYWORDS:
research software engineering; research software definition; research software community

TO CITE THIS ARTICLE:
Sochat V, May N, Cosden I, Martinez-Ortiz C, Bartholomew S 2022 The Research Software Encyclopedia: A Community Framework to Define Research Software. *Journal of Open Research Software*, 10: 2. DOI: *https://doi.org/10.5334/jors.359*

(1) OVERVIEW

INTRODUCTION

When you encounter a bear in the woods, you can be pretty sure that it's a bear. You might recognize features from childhood stories, the Discovery channel, or maybe even previous encounters. But then, what if someone asks you to sit down, and write a definition for a bear? How might you start? Well, you might be somewhat confident that it's a mammal, so you start with those features: having hair or fur, teeth, and being large. We don't need to work very hard because a lot of work has already gone into defining the features of a mammal. Without listing them all, the creature in question also needs to have sweat and mammary glands, three middle ear bones, a neocortex, and a four chambered heart. Now, even if we could make our bear friend transparent and see into him to answer these questions, we are again left with the same conundrum when we step up to the next level of evaluation – what makes a bear different from any other animal? What features are especially "beary"?

The same conundrum exists for research software. We have a strong sense of what constitutes software – it is some kind of compiled or interpreted program that is run by a computer. But then, what distinguishes "research software" from all other software? And further, do we really care about a set of exact attributes, or are we interested in amassing some significant number of general features? To return to our previous example with bears, I might step back and decide that I care less about identifying the bear, but rather, identifying a creature that might present some danger to me. This changes our way of thinking entirely because instead of thinking about ear bones and mammary glands, we start to consider size, aggression, presence of teeth, and arguably much more useful features in the context of our use case. This brings us to the idea that context is important. If I care about finding an animal to train for a honey commercial, my criteria will be very different than if I care about identifying a beast that might eat me for dinner.

This kind of context is equally important when we discuss research software, as the needs of a group or individual clearly frame any subsequent evaluation. Although efforts such as FAIR [1] exist to ensure that software in the research domain is findable, accessible, interoperable, and reusable (FAIR), and there is work to define the life-cycle [2] or measuring of such software [7], these efforts focus on quality or best practices, which is a different task than definition. There is also often an implied bias that the definition is self explanatory, and that research software is simply software that is used in research [3, 6]. However, the missing component to these efforts is that definition depends on context. Definitions of research software for a specific purpose like applying for a grant or submitting to a journal are typically interested in a subset of software. As an example, we can look at definitions of research software in our research software engineering community to understand why context-specific definitions are important. *The Journal of Open Source Software* (JoSS), for example, defines research software as

> software that: solves complex modeling problems in a scientific context (physics, mathematics, biology, medicine, social science, neuroscience, engineering); supports the functioning of research instruments or the execution of research experiments; extracts knowledge from large data sets; offers a mathematical library, or similar.

This domain-oriented definition would have a hard time including more general software such as application programming interfaces (APIs), supporting code for machine learning models, or databases. We can further look at a sample of rejected papers from JoSS to understand what was not considered research software. As of the writing of this paper, there are 319 rejected papers out of 3,584, and reviewing the first sample of 25 papers, 18 are rejected due to "Not substantial scholarly effort," 5 for "Does not meet definition of research software," and 1 for each of a "Desk rejection" and "Minor utility category." While this is a small sample, the fact that papers can be rejected for these reasons suggests that the authors did not fully understand the submission criteria, or perhaps that the definition can be vague or subjective, or too narrow. If we look across all rejected papers, a total of 83 of the software reviews on GitHub (issues) have some mention of "research software" suggesting that it came up as part of the discussion. If we inspect a crowd-sourced definition [8] of research software engineering, we find that research software engineers work on a set of tools for

> reproducibility, reusability, and accuracy of data analysis and applications created for research.

This suggests a different kind of focus on the goals of the software than criteria such as lines of code or domain. Finally, an even more generic definition comes from *IGI Global*, stating that research software is

> A computer-based application that converts inputs into outputs to support the user in one or more research tasks.

The issue arises when we need to define research software in the context of a specific goal. There are several contexts under which we might find ourselves in a position of needing to define a piece of software as research software (or not):

- **Funding bodies:** If a funding body is evaluating software to determine who receives a grant, they would clearly need to have a definition. There cannot

be any gray area about what constitutes research software, and what does not.

- **Journals:** Journals have traditionally been the means to share academic progress, and as software has been more acknowledged as an important part of research, we now see journals or sections of journals explicitly for research software. However, whether it's conscious or not, most journals likely have some non-trivial or (externally appearing) subjective way to classify something as research software. Journals need to have transparency in these criteria, and the scope of research software they consider.
- **People:** In that developing research software is a core part of many individuals' identities, having a definition is important to them.
- **Universities and National Labs:** In that these institutions conduct research that is empowered by research software, and need to employ and provide career perspectives for research software engineers and make funding and policy decisions for research, it's essential that they be able to define it.
- **Technology companies & startups:** Firms commonly conduct software-based research towards their own business goals, often using tools developed in an academic environment and potentially feeding back to the wider community for example with insights and new, or improved, tools. These businesses would benefit from a definition for research software for instance to distinguish from other software they use or are responsible for, e.g. as a product.

While any particular context-driven definition is not inherently wrong, given the diversity of these different contexts and categories we suggest that the approach to define a single definition of research software is challenging if not impossible. Groups, organizations, or journals that need to more clearly communicate about a definition for research software need a different approach, as with the current approach authors might spend unnecessary time preparing a submission that is deemed to not fit the hard-to-understand definition. While there are currently efforts that eventually will discuss a definition for research software (e.g., The RD Research Alliance [1]), arguably an effort that is fully community driven, open, and has international asynchronous participation on GitHub would help to guarantee that a diversity of opinions across domains of science (e.g., life sciences, social sciences, digital humanities) are taken into account. This is the rationale behind this work for the Research Software Encyclopedia.

A Community-driven Approach

The Research Software Encyclopedia is a community driven, open source strategy that takes a different approach. Instead of trying to provide a single definition for research software, it provides a method and framework to go about evaluating software in the context relevant to a particular need. By providing lists of criteria and a taxonomy of domains, a user can make a context-specific choice about a definition of research software. While this choice might be subjective, the criteria and categorization provided by this framework are not, making it easy for an individual to evaluate their software on the different categorizations, and then easily map to a context of interest. The remainder of this document will discuss the design and implementation of the software.

IMPLEMENTATION AND ARCHITECTURE

The Research Software Encyclopedia has several different tools and databases with core tools implemented in Python, with data storage in JSON, and web interfaces that use JavaScript, HTML, and CSS styling. These languages and technologies were chosen as they are well known in scientific programming, and would be easy for research software engineers to contribute to. The components include an explicit framework or algorithm to assess a piece of research software, a means to filter criteria points or categories (a taxonomy) for a given use case, and automation and web interfaces for interaction with a community database.

Criteria for Research Software

The creation of criteria to define research software was a community effort that took several iterations, and took an approach to iterate over simple questions to ask any piece of software such as defining creators, goals, licensing, citation, and intended users. While the details are out of scope for this meta-paper, the document is available for the interested reader [9]. The final set of questions or criteria for the Research Software Encyclopedia were:

- Is it software (all research software must be software) (yes/no)
- Is it used by at least one researcher? (yes/no)
- Has it been cited in a research context? (yes/no)
- Is it intended for a particular scientific domain? (yes/no)
- Would taking it away be a detriment to research? (yes/no)
- Was it created with the intention to be used for research? (yes/no)

Any specific individual or group could use these questions to derive a meaningful definition of research software for their needs, and the questions would need to be answered only once for any piece of software to be useful in many different contexts.

A Taxonomy of Research Software

If it's the case that we have a general definition for research software that is based on its intention, users, and impact on the research space (the criteria described

in the previous section), we need to allow for a user of the definition to scope his or her definition to some subset. We need to be able to further break research software into sub-groups, and thus empower people to refer to some subset. This calls for a taxonomy of research software, which was also developed via a community effort in the document previously linked.

- Software to directly conduct research
  - Domain specific software
    - Domain-specific hardware (e.g., software for physics to control lab equipment, or embedded hardware)
    - Domain-specific optimized software (e.g., neuroscience software optimized for GPU)
    - Domain-specific analysis software (e.g., SPM, fsl, afni for neuroscience)
  - General software
    - Numerical libraries (includes optimization, statistics, simulation, e.g., numpy)
    - Data collection (e.g., web-based experiments or portals)
    - Visualization (interfaces to interact with, understand, and see data, plotting tools)
- Software to support research
  - Explicitly for research
    - Interactive development environments for research (e.g., Matlab, Jupyter)
    - Workflow managers
    - Provenance and metadata collection tools
  - Used for research, but not explicitly for it
    - Databases
    - Application programming interfaces
    - Frameworks (to generate documentation, content management systems, etc.)
  - Incidentally used for research
    - Operating Systems
    - Package Managers
    - Virtualization technologies
    - Formatting, indexing, or other small helper libraries
    - Scheduling and task management (for people)
    - Version Control
    - Text Editors and Integrated Development Environments (IDEs)
    - Communication tools or platforms (e.g., email, video-conferencing, etc.)
    - Infrastructure (e.g., on-prem or cloud servers used for services or research needs)
    - Testing or software libraries

Note that a piece of software that ultimately might not be considered research software (e.g., the operating system Linux or version control software git) can still be classified here, as it is incidentally used for research.

To make these criteria and taxonomy programmatically accessible, a library *rseng* is provided that defines both criteria and the taxonomy in YAML, and makes them easily loadable into Python dictionaries for interested researchers to develop with, and provides functions to export to JSON or csv, or generate markdown to render into a web interface. The web interface with markdown files for the current criteria and taxonomy is provided by the *GitHub Pages branch* of the same repository, and available at *https://rseng.github.io/rseng/*. This pairing is done so that documentation and code live alongside one another. A researcher could easily use or extend this work to create, visualize, and programatically provide their own set of taxonomy and criteria items. The library is *available on Pypi*. Along with providing human-friendly user interfaces, the *taxonomy* and *criteria* site also provides an application programming interface (API) that always makes available the most recent taxonomy and criteria for other services such as the Research Software Encyclopedia to use, discussed next.

The Research Software Encyclopedia

The Research Software Encyclopedia is Python software that provides a command line tool to create and manage a custom database of research software. It is also *available on Pypi*, has documentation rendered on GitHub pages alongside the source code and published at *https://rseng.github.io/rse/*, and has source code on GitHub at *https://github.com/rseng/rse*. More specifically, the software includes:

- Commands to add or remove or list software
- Parsers for remote software repositories (e.g., GitHub, GitLab)
- Scrapers to discover new software repositories via resource APIs
- A criteria and taxonomy annotation interface for a software database
- Containers for pre-built environments to use the software
- An application programming interface for the database
- Annotation via the command line or a web interface

A quick example of installing the software and creating a database of research software with two entries from GitHub might look like the following:

```
$ pip install rse[all]
$ rse init
$ rse add https://github.com/singularityhub/singularity-hpc
$ rse ls
1 github/singularityhub/singularity-hpc
$ rse add https://github.com/dask/dask
$ rse ls
```

```
1 github/dask/dask
2 github/singularityhub/singularity-hpc
$ tree database
database
         github
               dask
                  dask
                     metadata.json
         github
              singularityhub
                  singularity-hpc
                     metadata.json
```

The repository above is an example of a flat file database, which can be pushed to GitHub to work on collaboratively. An interested user that wants a more production (e.g., relational) database can simply *configure* their Research Software Encyclopedia to use one. After creating the database, the user might then be interested in *annotation*, or more general sharing of the software. This is made possible by way of the export command and automation, discussed next.

The Community Software Database

Given an easy means to manage a flat file database on GitHub, the Research Software Encyclopedia combines its relevant components into a community software database available at *https://github.com/rseng/software*, which is generated with an export command that considers the configuration file, host, and other relevant parameters:

```
#!/bin/bash
export RSE_HOST=https://rseng.github.io
export RSE_URL_PREFIX=/software/
export RSE_CONFIG_FILE=rse.ini
rse export --type repos-txt repos.txt --force
rse export --type static-web docs/
```

The commands above can generate an entire static interface for a database, and via automation the Research Software Encyclopedia can discover new software and update itself weekly. This update is done via a scheduled GitHub workflow that uses the rse software "scraper" functionality to look for new software from the *Journal of Open Source Software*, *bio.tools*, the *Hal Research Software Database* [3], the *Research Software Directory* [4], and *ROpenSci* [5]. Other scrapers can easily be added on request. By itself, this single repository will provide a single source of data for a researcher interested in studying research software, as defined by the different journals and groups that are scraped.

Along with providing the software in an *community software interface*, an interested researcher can select any piece of software to annotate in the browser directly, which will then open a pre-populated GitHub issue. The GitHub issue will then be automatically labeled, and the labeling triggers a workflow to save the annotation to the database, and close the issue when it is complete. Thus, all the annotations live as flat csv files alongside the software data. Interesting pieces of software that are added are written about in the *Software Showcase* and shared on social media.

QUALITY CONTROL

For the Research Software Encyclopedia, along with the taxonomy and criteria repository, and software database, tests are run via continuous integration for each pull request or merge into the main branch. For the Research Software Encyclopedia command line software, tests include testing the functionality and output of parsers (e.g., GitHub, GitLab), along with ensuring that all commands produce expected output. The criteria and taxonomy repository and tool also tests the functionality of the main commands, and output contents. For the software database, the quality control comes from the sources. For example, papers and associated software that goes into JoSS goes under a formal review process, and the other software databases are curated by teams of research software engineers.

(2) AVAILABILITY

OPERATING SYSTEM

The Research Software Encyclopedia should work on most Unix and Linux flavored distributions, or those that can run Docker. The software was developed on Ubuntu 18.04 and 20.04.

PROGRAMMING LANGUAGE

The Research Software Encyclopedia set of tools supports Python 3.5 and higher. Python 2.x is not supported.

ADDITIONAL SYSTEM REQUIREMENTS

To interact with a relational database (e.g., MySQL or Postgres) the system would need to install the database software natively, or run via a Docker container.

DEPENDENCIES

The Research Software Encyclopedia requires the Python requests library for basic function, several Flask libraries for advanced use of the interface, and pytest for testing. See the repository version.py file for details.

LIST OF CONTRIBUTORS

All authors contributed to the development of criteria and taxonomy items. Vanessa Sochat is the primary author of the software, documentation, and interfaces.

SOFTWARE LOCATION

Name: rse-0.0.34.tar.gz

Persistent identifier: *https://zenodo.org/record/5546046#.YVi-uXtMFH4*

Package manager: *https://pypi.org/project/rse/0.0.34/*
License: MPL 2.0
Publisher: Vanessa Sochat
Version published: 0.0.34
Date published: January 28, 2022

Criteria and Taxonomy Archive pypi is used as a package manager for releases.

Name: rseng-0.0.18.tar.gz
Persistent identifier: *https://zenodo.org/record/5546052#.YVi-43tMFH4*
Package manager: *https://pypi.org/project/rseng/0.0.18/*
License: MPL 2.0
Publisher: Vanessa Sochat
Version published: 0.0.18
Date published: December 6, 2020

Research Software Encyclopedia Code repository GitHub

Name: *https://github.com/rseng/rse*
Persistent identifier: *https://zenodo.org/record/5546054#.YVi_FHtMFH4*
License: MPL 2.0
Date published: November 27, 2020

Taxonomy and Criteria Code repository GitHub

Name: *https://github.com/rseng/rseng*
Persistent identifier: *https://github.com/rseng/rseng*
License: MPL 2.0
Date published: September 28, 2020

LANGUAGE

The software is implemented in Python, with supporting scripts in bash for testing, and configuration files in yaml.

(3) REUSE POTENTIAL

On a high level, a framework to define research software allows us to have different definitions of research software useful for different purposes. This allows us to treat research software differently depending on what is our objective, and to have clear guidelines to decide if a specific piece of software is or is not research software. For example, we could have a definition of what we consider research software for short term software preservation and a different definition of research software for long term software preservation.

The general nature of the Research Software Encyclopedia, and the availability in several different components (the criteria and taxonomy, the database manager and the database itself) make it reusable for a wide variety of needs not described here. The shared community databases can be used to provide an automatically generated set of research software, as identified by journals and databases that publish it, either for further analysis of the software or change over time. The criteria and taxonomy items alone can be used by a funding body to easily define criteria or categories for research software, and the software for these definitions can be used for a completely different set of criteria or taxonomy items. A user or specific domain could also use the command line database manager to create a domain or topic specific database of their own software, either as a personal portfolio or for a group such as a lab. Finally, the annotation interface can provide data for a more substantial research project to understand software, or adopted to provide an annotation interface for something else entirely. All code and interfaces are open source on GitHub, and contributions and ideas are welcome.

CONCLUSION

In this paper, we have discussed criteria for research software, a taxonomy to define it, and a general framework and tools for creating a definition useful for a particular context. The biggest insight to this process is that, like many things, there is not a one-size-fits-all answer. Despite this quality, we still need to be able to make classifications that drive life decisions. The definition of research software is, somewhat ironically, not a clear definition that you write on a single page, but rather a gradient of features that can be filtered and viewed based on the context they are viewed. The definition of research software is subjective on the level of a use case, but not subjective in terms of its overall assessment. Although we might never come to an agreed upon definition, we can be somewhat confident in our ability to ask a series of questions about some piece of software, and then decide which questions and responses are important for our definition. We can be confident that although we might not completely understand research software, by starting a simple taxonomy and criteria, we can further develop machine learning or other data science projects to improve our understanding. This kind of work will not only support researchers that use research software, but also empower the research software engineers that create it.

ACKNOWLEDGEMENTS

The authors would like to thank the community for feedback on the draft, and the software.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Vanessa Sochat *orcid.org/0000-0002-4387-3819*
Computer Scientist, Lawrence Livermore National Lab, US
Nicholas May *orcid.org/0000-0002-1298-1622*
Ian Cosden *orcid.org/0000-0003-3780-9172*
Director Research Software Engineering, Princeton University, US
Carlos Martinez-Ortiz *orcid.org/0000-0001-5565-7577*
Netherlands eScience Center, NL
Sadie Bartholomew *orcid.org/0000-0002-6180-3603*
Computational Scientist, National Centre for Atmospheric Science & University of Reading, GB

REFERENCES

1. **Daniel SK, Morane G, Tom H.** Taking a fresh look at FAIR for research software; 2021. DOI: *https://doi.org/10.1016/j.patter.2021.100222*
2. **Gomez-Diaz T, Recio T.** On the evaluation of research software: the CDUR procedure. *F1000Res*. 2019; 8: 1353. DOI: *https://doi.org/10.12688/f1000research.19994.1*
3. **Di Cosmo R, Gruenpeter M, Marmol B, Monteil A, Romary L, Sadowska J.** Curated Archiving of Research Software Artifacts: lessons learned from the French open archive (HAL); 2019. *https://hal.archives-ouvertes.fr/hal-02475835*. DOI: *https://doi.org/10.2218/ijdc.v15i1.698*
4. **Spaaks JH, Klaver T, Verhoeven S, Maassen J, Pawar P, van Hage W, Ridder L, Kulik L, Bakker T, van Hees V, Bogaardt L, Mendrik A, van Es B, Attema J, Ranguelova E, van Nieuwpoort R.** Research Software Directory (1.2.0). *Zenodo*; 2020. DOI: *https://doi.org/10.5281/zenodo.3631783*
5. **Boettiger C,** et al. Building software, building community: lessons from the rOpenSci project. *Journal of open research software 3.1*; 2015. DOI: *https://doi.org/10.5334/jors.bu*
6. **Maimone C.** Supporting Research Software and Research Software Engineers; Oct. 2019. *https://sites.northwestern.edu/researchcomputing/2019/10/07/supporting-research-software-and-research-software-engineers.* Accessed 2 Oct 2021.
7. **Hong NPC.** Why do we need to compare research software, and how should we do it? 2016. *http://ceur-ws.org/Vol-1686/WSSSPE4_paper_29.pdf*.
8. **Research Software Engineering.** Oct. 2021. *https://en.wikipedia.org/wiki/Research_software_engineering.* Accessed 2 Oct 2021.
9. **The Research Software Encyclopedia.** Dec. 2020. *https://docs.google.com/document/d/1wDb0udH9OrFWrMBsAVb8RrUMCKKRHoyEep7yveJ1d0k/edit.* Accessed 2 Oct 2021.

TO CITE THIS ARTICLE:
Sochat V, May N, Cosden I, Martinez-Ortiz C, Bartholomew S 2022 The Research Software Encyclopedia: A Community Framework to Define Research Software. *Journal of Open Research Software*, 10: 2. DOI: *https://doi.org/10.5334/jors.359*

Submitted: 01 December 2020 Accepted: 25 February 2022 Published: 04 March 2022

Journal of Open Research Software is a peer-reviewed open access journal published by Ubiquity Press.