**Fachbeitrag**

Stephan Druskat, Oliver Bertuch, Alexander Struck

# Towards Research Software-ready Libraries

## Forschungssoftware in Bibliotheken

https://doi.org/10.1515/abitech-2023-0031

**Abstract:** Software is increasingly acknowledged as valid research output. Academic libraries adapt to this change to become research software-ready. Software publication and citation are key areas in this endeavor. We present and discuss the current state of the practice of software publication and software citation, and discuss four areas of activity that libraries engage in: (1) technical infrastructure, (2) training and support, (3) software management and curation, (4) policies.

**Keywords:** research software, software citation, software publication

**Zusammenfassung:** Software wird zunehmend als gültiges Forschungsergebnis anerkannt. Wissenschaftliche Bibliotheken passen sich diesem Wandel an, um für Forschungssoftware gerüstet zu sein. Softwarepublikation und -zitierung sind hierbei Schlüsselbereiche. Wir präsentieren und diskutieren hier den aktuellen Praxisstand und heben vier Bereiche hervor, in denen Bibliotheken aktiv werden können, um für Forschungssoftware gerüstet zu sein: (1) technische Infrastruktur, (2) Schulung und Support, (3) Management und Kuratierung von Software, (4) Richtlinien.

**Schlüsselwörter:** Forschungssoftware, Softwarezitierung, Softwarepublikation

## 1 Introduction

In research, software fulfills two general roles: It is a method to yield or enable research results, for example through creating, measuring, processing, analyzing or visualizing data, but also through modeling and simulation. Software is also an outcome of research that is developed, implemented, and maintained. As such, it embeds research knowledge[1] and represents complex theoretical constructs that cannot be easily described in a paper.[2] Therefore, research software must be defined as a research output in its own right.[3] This is increasingly acknowledged across research communities.[4]

Libraries in general, and academic libraries in particular, are responsible for recording and attesting research output. Under this mandate, libraries are therefore also responsible for recording and attesting research software, as a central building block to support scientific reporting and improved findability and reuse, e. g., through their indexing/cataloging efforts.

This responsibility is directly derived from the requirements of scientific reporting today, that call for "extensive documentation of the hardware, software and input data"[5] and for making accessible "data [...], software, model parameters, workflow information, theoretical calculations, and tests of the analysis and/or software"[6]. In order to fulfill the requirement of making software accessible, libraries support the practice of software publication. And in order to enable the documentation of software and its use, libraries enable software citation, e. g., through safeguarding metadata quality, but also offer support via training services.

1 Anzt, Hartwig et al. "An Environment for Sustainable Research Software in Germany and beyond: Current State, Open Challenges, and Call for Action." *F1000Research* 9 (January 26, 2021): 295. doi:10.12688/f1000research.23224.2.

2 Jay, Caroline et al. "The Challenges of Theory-Software Translation" (F1000Research, October 2, 2020), doi:10.12688/f1000research.25561.1.

3 Jay, Caroline, Robert Haines, Daniel S. Katz. "Software Must Be Recognised as an Important Output of Scholarly Research." *International Journal of Digital Curation* 16,1 (April 26, 2021): 6. doi:10.2218/ijdc.v16i1.745.

4 Smith, Arfon M. et al. "Software Citation Principles." *PeerJ Computer Science* 2,e86 (2016). doi:10.7717/peerj-cs.86; Doerr, Allison et al. (eds.). "Giving Software Its Due." *Nature Methods* 16,3 (March 2019): 207–207. doi:10.1038/s41592-019-0350-x.

5 Borgman, Christine L., Jillian C. Wallis, Matthew S. Mayernik. "Who's Got the Data? Interdependencies in Science and Technology Collaborations." *Computer Supported Cooperative Work (CSCW)* 21,6 (August 2012): 489. doi:10.1007/s10606-012-9169-z.

6 Weitz, David A. et al. "NSF Workshop Systematic Approach to Robustness, Reliability, and Reproducibility in Scientific Research." Beckman Center of the National Academy of Sciences and Engineering University of California, Irvine (2017): 11. Last checked 04.07.2023. http://www.mrsec.harvard.edu/2017NSFReliability/.

In software publication, authors publish their software together with relevant metadata (see section 2). In software citation, authors that report research (results) also cite software as they would cite a paper[7] and in such a way that allows the unique identification of a software version that was used in the reported research. Software citation must also enable access to that software version in an archive or source code repository.

Software publication is integral to research software practice:

1. It provides evidence of methods that were applied in research. This is important to enable the reproducibility of research results.
2. It provides identification and traceability for research outcomes that are software. This is important for the identification of previous work and provenance of research results.
3. It records the software work that has been carried out in research. This is important to evaluate the creation of value in research processes.

These functions of software publication are activated through the practice of software citation (see section 3). Citation of software publications links research results to applied methods, identifies software as part of the provenance of research results, and provides attribution and credit to software authors. Software citation can also enable better software sustainability through incentivizing sustainable development and advertising reuse.[8]

In this paper, we describe how software publication and software citation in combination allow libraries to fulfill their mandate to record and attest research outputs with regard to research software. We also describe areas of activity for libraries to engage in that support their own work with research software, as well as the work of their users. In short, we lay out a path for libraries to become "research software-ready" in a way that supports the FAIR Principles for Research Software (*FAIR4RS*[9]).

In the following sections, we describe the basics of software publication (2) and software citation (3), as well as existing challenges. We then establish requirements for libraries to become research software-ready (4) and activities that support this endeavor. Finally, we conclude with a summary (5).

## 2 Software Publication

We define software publication as the process of depositing software (versions) and metadata in a *publication repository*, where the deposit is uniquely identified with a machine-resolvable persistent identifier (PID), such as a Digital Object Identifier (DOI). This is in contrast to other practices sometimes also termed "software publication" (see section 2.1). We understand software publication as an active process, similar to other forms of research reporting, e. g., paper publication. In this process, the recording and provision of curated, rich metadata is the main factor that supports FAIR research software[10] and enables software citation. Software publication records consist of the machine-readable metadata pertaining to the published software (version), as well as any archived software source code or (executable) artifact snapshots of the version. Minimally, these metadata must include the information necessary for software citation:[11]

- Unique identifier
- Software name
- Author(s)
- Version identifier[12]
- Publication/release date
- Location/*source code repository*

Further metadata should be supplied to increase FAIRness and better support the evaluation of software for reuse. These include for example license and legal information, high-level descriptions, research domains, and software-specific metadata such as programming languages, dependency information, runtime requirements, and software engineering metrics.[13] Domain-specific repositories, digital libraries or archives may suggest additional metadata, such as preferred citation method (e. g. in http://ascl.net) or classifications (e. g. in https://swmath.org/). Software metadata standardization efforts include the Citation File Format (CFF[14]) and CodeMeta[15] and are documented

7 Smith et al. 2019.

8 Druskat, Stephan, Daniel S. Katz, Ilian T. Todorov. "Research Software Sustainability and Citation." *2021 IEEE/ACM International Workshop on Body of Knowledge for Software Sustainability (BoKSS)* (2021): 1–2. doi:10.1109/BoKSS52540.2021.00008.

9 Chue Hong, Neil P. et al. "FAIR Principles for Research Software (FAIR4RS Principles) (1.0)." *Research Data Alliance* (May 24, 2022). doi:10.15497/RDA00068.

10 Chue Hong 2022.

11 Smith et al. 2019: 6.

12 While it is good practice to provide a version identifier following the software project's versioning scheme, it is not strictly required if the unique identifier resolves to a specific version (see Katz et al. "Recognizing the value of software: a software citation guide." *F1000Research* 9 (12 January 2021): 1257. doi:10.12688/f1000research.26932.2.).

13 Druskat, Stephan et al. "Software Publications with Rich Metadata: State of the Art, Automated Workflows and HERMES Concept." *ArXiv* (January 22, 2022). doi:10.48550/arXiv.2201.09015.

14 Druskat, Stephan et al. "Citation File Format." *Zenodo* (August 2021). doi:10.5281/ZENODO.5171937.

15 Jones, Matthew B. et al. *CodeMeta: An Exchange Schema for Software Metadata. Version 2.0* (2017). doi:10.5063/schema/codemeta-2.0.

in Christopherson et al.[16] and Park and Wolfram.[17] To make these metadata consumable for humans, they are often presented on landing pages for each software publication record.

The inclusion of software source code and/or artifacts in software publications is optional and depends on the software governance and license. For *open source software*, the source code and software metadata should be published in a fully open access publication repository. The source code for *inner source software* should be published in a publication repository with full accessibility for members of the respective community, e. g., within the institution in which the source code may be shared. Additionally, even if the software source code is not accessible to the general public, the respective software metadata should still be published open access to satisfy the FAIR4RS principles[18] and to enable software citation. *Closed source software* may not be made publicly accessible due to legal reasons or export restrictions, yet the software metadata should be published openly in the same way as for *inner source software* (see section 2.1).

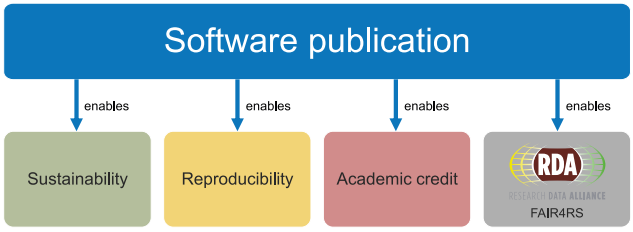


**Fig. 1:** Software publication enables sustainability, reproducibility, academic credit, and adheres to the FAIR4RS principles[19] (adapted from Druskat et al.[20] under CC BY-4.0 Intl.)

Software publication as described above facilitates potentials for software sustainability, reproducibility, and academic credit for software authors. It also asserts adherence to the FAIR4RS principles.

Software publication facilitates software sustainability by producing and providing archived records of software versions together with metadata that describes them. This descriptive long-term provision increases the "capacity of the software to endure"[21]. Finding published software records and being able to access and understand them would make it possible to use the software and adapt it for new use cases. Although the availability of software alone is not enough to make it sustainable, software that is not available cannot possibly be sustainable.

Published software records establish the basic building block to achieve "computational reproducibility"[22] of research by providing methods applied in the original research. Again, the availability of software alone is not enough to enable even re-runability[23] of the software, but unavailable or undescribed software impedes reproducibility. The re-runability of software can be improved by publishing it with metadata pertaining to the complete runtime environment as used during the research to be reproduced,[24] or containers preserving the environment itself.[25]

Author information is an integral part of software publication metadata in the same way it is for other research output formats such as papers or books (see section 3). Therefore, software publication also transitively enables academic credit for software authors, as the complete and correct author metadata is made available for software citation (see section3).

Finally, published software metadata, either by itself or published alongside software source code or artifacts, contributes to compliance with the FAIR4RS principles.[26]

## 2.1 How is Software (Not) Published?

Unlike the publication of textual research outputs, which has been in practice for centuries, software publication as defined above is a relatively new concept. It is therefore not

16 Christopherson, Allan et al. "Software Metadata Recommended Format Guide." Cornell University Library, 2022. doi:10.7298/XE9S-3B15.

17 Park, Hyoungjoo, Dietmar Wolfram. "Research Software Citation in the Data Citation Index: Current Practices and Implications for Research Software Sharing and Reuse." *Journal of Informetrics* 13,2 (May 1, 2019): 576. doi:10.1016/j.joi.2019.03.005.

18 Chue Hong et al. 2022.

19 Chue Hong et al. 2022.

20 Druskat et al. 2022.

21 S. Katz, Daniel. "Defining Software Sustainability." *Daniel S. Katz's Blog*, September 13, 2016. Last checked 04.07.2023. http://web.archive.org/web/20221012111742/https://danielskatzblog.wordpress.com/2016/09/13/defining-software-sustainability/.

22 National Academies of Sciences, Engineering, and Medicine. *Understanding Reproducibility and Replicability.* National Academies Press (US), 2019. Last checked 04.07.2023. https://www.ncbi.nlm.nih.gov/books/NBK547546/.

23 Benureau, Fabien C. Y., Nicolas P. Rougier. "Re-Run, Repeat, Reproduce, Reuse, Replicate: Transforming Code into Scientific Contributions." *Frontiers in Neuroinformatics* 11 (2018). Last checked 04.07.2023. https://www.frontiersin.org/articles/10.3389/fninf.2017.00069.

24 The Turing Way Community. "The Turing Way: A Handbook for Reproducible, Ethical and Collaborative Research." July 27, 2022. doi:10.5281/zenodo.7625728.

25 Nüst, Daniel et al. "Ten Simple Rules for Writing Dockerfiles for Reproducible Data Science." *PLOS Computational Biology* 16,11 (October 11, 2020): e1008316. doi:10.1371/journal.pcbi.1008316.

26 Chue Hong et al. 2022.

surprising that a "general lack of active preservation"[27] of research software has been observed.

The current insufficient practice of software publication leads to negative effects for software citability, findability, and accessibility.[28] Instead of a formal reference to the software itself (see section 3, Smith et al.[29]), software is often only either informally mentioned or a paper is cited instead of the software in the research literature.[30] Additionally, software "is frequently inaccessible (15 %–29 % of packages in any form; between 90 % and 98 % of specific versions […])"[31] which impedes peer review and replication efforts. This situation also leads to a lack of possibility for research software authors to receive credit for their work.

To alleviate this situation and bridge the gap between the traditional system of text publication, journals dedicated to publishing peer-reviewed software have been launched in recent years, for example, *Journal of Open Research Software (JORS)*,[32] *Journal of Open Source Software (JOSS)*,[33] and *SoftwareX*.[34] These *software journals* offer a traditional workflow for publishing a non-traditional research output as *software paper*, and in some cases (most notably *JOSS*) use peer review to assess the engineering and usability quality of the software and documentation. At the same time, software journals are conceptually unable to cater for a core characteristic of software as a research output. They record and describe – through an accompanying text document not unsimilar to a traditional short paper – a *single version* of a software. This may be sufficient for those software types that have exactly one specific use case and for which only one version exists (see section 2.2), but insufficient for software for which many versions may be produced over their lifecycle. Software papers almost instantaneously expire, as they become out-of-date as soon as the next iteration of the published software is produced.[35] This may lead to incorrect citations of a software version if the version actually used in research is not the software paper version, especially if software authors give the software paper as preferred citation of the software. Incorrect citations, in turn, make attempts to reproduce reported research much harder. Despite this central shortcoming, software journals currently represent a useful bridge technology that pioneered an initial implementation of external software peer review.

Software has also been "published" under definitions differing from our own (see section 2). While we do not consider these practices *software publication*, it is useful to briefly name them here for differentiation.

A common way of publicizing software is by publishing a paper that describes the software. This has many disadvantages in comparison to real software publication:

1. Papers that describe software are paper publications, not publications of software itself. This is insufficient under the basic assumption that software is a valid research output in its own right, on par with academic articles.[36]
2. Papers that describe software may describe a specific version of the software but cannot describe future versions. Thus, they are outdated as soon as development of the described software continues.

Sometimes, developing software publicly is confused with publishing software. Platforms such as GitHub[37] or GitLab[38] facilitate public access to source code and enable contributions to the source code by others. They also allow for the creation of *releases*: single source code snapshots can be marked (*tagged*) as being of a specific stability, quality, or level of completeness that warrants the distribution to users. These snapshots can be promoted to "releases" which can have their own landing page with, e. g., a description of the release or changes in comparison to past releases. However, these snapshots are not frozen as they would be in an archive. Releases can be deleted, renamed, and re-done so that their identifiers (URLs) may change. Also, whole repositories made publicly accessible on the mentioned platforms can be renamed and deleted, and often lack necessary metadata as required in Smith et al.[39] (see section 2). Given this situation, publicly accessible source code, or releases of the same, cannot be considered software publications.

**27** AlNoamany, Yasmin, John A. Borghi. "Towards Computational Reproducibility: Researcher Perspectives on the Use and Sharing of Software." *PeerJ Computer Science* 4 (September 17, 2018): 19. doi:10.7717/peerj-cs.163.

**28** Struck, Alexander. "Research Software Discovery: An Overview." *2018 IEEE 14th International Conference on E-Science (e-Science)* (2018): 33–37. doi:10.1109/eScience.2018.00016.

**29** Smith et al. 2019.

**30** Druskat, Stephan et al. "Don't Mention It: Challenges to Using Software Mentions to Investigate Citation and Discoverability." *PeerJ Computer Science*, forthcoming.

**31** Howison, James, Julia Bullard. "Software in the Scientific Literature: Problems with Seeing, Finding, and Using Software Mentioned in the Biology Literature." *Journal of the Association for Information Science and Technology* 67,9 (May 13, 2015): 2137. doi:10.1002/asi.23538.

**32** Last checked 04.07.2023. https://openresearchsoftware.metajnl.com/.

**33** Last checked 04.07.2023. https://joss.theoj.org/.

**34** Last checked 04.07.2023. https://www.journals.elsevier.com/softwarex.

**35** Similarly, software papers cannot represent versions older than the published one, although they may still be used in research.

**36** Smith et al. 2019.

**37** Last checked 04.07.2023. https://github.com.

**38** Last checked 04.07.2023. https://about.gitlab.com.

**39** Smith et al. 2019: 6.

## 2.2 Challenges in Software Publication

Research software publication as defined above (section 2) includes the deposition of software version metadata and artifacts in a *publication repository*, and the creation of persistent identifiers for the deposits. Research software, however, is very heterogeneous: it is developed and used in different research domains for different use cases and uses a variety of technologies and programming languages. It is developed by people with different domain and professional backgrounds and levels of experience, within distinctive contexts and with different aims.[40] This heterogeneity has an impact on how the software is being developed, its quality and maturity, and ultimately its mode of publication.

Different models exist to provide classes for research software and thus make it easier to understand the different subsets that exist. Application classes, as put forward in institutional software engineering guidelines,[41] cluster research software based on software quality that reflect the needs of stakeholders. The Australian Research Data Commons' National Agenda for Research Software[42] defines three interdependent types of research software based on their purpose: (A1) *analysis code* (or *data-coupled code*) that captures research processes and methods; (A2) *prototypes* that showcase a new idea, method, model or algorithm; and (A3) *research software infrastructure* that provides established ideas, methods or models.

Using similar terminology with a stronger focus on concrete use cases, Felderer et al.[43] suggest (B1) *modeling, simulation and data analytics [software]* including data-centric uses such as data science/engineering/assimilation, (B2) *proof-of-concept software prototypes* specifically in engineering science, and (B3) *infrastructure and platform software*, specifically research data and software management systems. Additionally, they add (B4) *embedded control software* for physical and chemical experiments and instruments, which arguably has specific requirements, e. g., in terms of safety and security, that may impact the software development and maintenance process.

Hinsen,[44] in contrast, separates different layers of research software in a "scientific software stack" based on their abstractive distance from computing hardware and operating system, but also by application scope: (C1) *non-scientific infrastructure* includes general purpose software such as compilers, interpreters, software libraries, and data management; (C2) *scientific infrastructure* includes infrastructure software for domain-independent scientific purposes such as mathematical libraries, scientific data management, and visualization tools; (C3) *domain-specific tools* include software developed for application in a specific research domain; and (C4) *project-specific code* includes scripts, workflows or computational notebooks specific to a research endeavor.

If we abstract over these four exemplary models, we can say that in terms of software publication, different approaches apply depending on the broadness of the application scope and the level of maturity. Generally speaking, if the software has a broad application scope and high maturity, it is likely that a large number of versions are published. Additionally, such software is more likely to be engineered to a higher degree, and potentially more modularized, so that different modules may be published independently from each other and from their underlying framework. At the other end of the spectrum, the more narrow the application scope and the lower the maturity level is – for example in the case of a script for a specific analysis of a specific dataset – the more likely it is for the software not to be published at all, which is the fundamental challenge, or to be published in its complete context. In this case, a *mixed content publication* including code, data, and textual output may be more suitable, as the specificity of the software work is high and its size likely low. Similarly, there may only be a single publication of this complex *research object*,[45] especially when used as a *replication resource*.[46] It is obvious that determining the most suitable publication practice for a given research software project is non-trivial. It is important to note that despite this, research software must be published in all cases where it represents a research result itself, or has been used to produce research results.

**40** Hettrick, Simon et al. "International RSE Survey 2022." *Zenodo* (August 22, 2022). doi:10.5281/zenodo.7015772; Hasselbring, Wilhelm et al. "Open Source Research Software." *Computer* 53,8 (August 2020): 84–88. doi:10.1109/MC.2020.2998235.

**41** Schlauch, Tobias, Michael Meinel, Carina Haupt. "DLR Software Engineering Guidelines." *Zenodo* (August 2018). doi:10.5281/ZENODO.1344612; Bertuch, Oliver et al. "Guidelines for the Development and Distribution of Software at Forschungszentrum Jülich." (2022). Last checked 04.07.2023. http://hdl.handle.net/2128/33259.

**42** Australian Research Data Commons. "A National Agenda for Research Software." *Zenodo* (March 28, 2022). doi:10.5281/zenodo.6378082.

**43** Felderer, Michael et al. "Toward Research Software Engineering Research." *Zenodo* (June 9, 2023). doi:10.5281/zenodo.8020525.

**44** Hinsen, Konrad. "Dealing with Software Collapse." *Computing in Science & Engineering* 21,3 (2019): 104–108. doi:10.1109/MCSE.2019.2900945.

**45** Soiland-Reyes, Stian et al. "Packaging Research Artefacts with RO-Crate." *Zenodo* (August 13, 2021). doi:10.5281/ZENODO.5146228.

**46** Trisovic, Ana. "Cluster Analysis of Open Research Data: A Case for Replication Metadata." *International Journal of Digital Curation* 17,1 (2022): 13. doi:10.2218/ijdc.v17i1.833.

Another challenge inherent to software publication is quality assurance. Like software in general, research software is often developed continuously. In contrast to papers, software has no "final product" as such; it can be developed further and new versions can be released and published. Development may be highly dynamic, with new versions being released regularly, sometimes in short intervals. It is impossible to conduct quality assurance in the form of traditional peer review for each and all of these versions. This is true not just due to the quantity of the output but also to the depth in which software peer review would have to be undertaken. Such reviews not only would have to cover the scientific correctness and methodology for the representation of highly complex theoretical constructs and implementation of algorithms, the contributions to the state of the art, and (ideally) the readability of the source code, but also have to take into account the different dimensions of software engineering: requirements engineering, architecture and design, constructive quality, validation and verification, documentation, development workflows, unfavourable code patterns, security concerns, etc. Some software journals, most prominently *JOSS*, cover some of the latter in their review process. Beyond that, however, software reviews are embedded in software development as part of change management, if at all.[47] This is in stark contrast to traditional practice in text publication, where the burden of organizing peer review lies with the publisher – or rather their editors – not with the authors themselves or their collaborators.

A better practice of software publication requires the necessary technical infrastructure to be available, and publication procedures to be established as standard or default. This is not yet the case. In terms of infrastructure, not all *publication repositories* are ready to ingest and represent software, for example because they miss a software record type in their data model, or cannot represent software metadata well. As for procedures, there is not yet a commonly accepted definition of software publication (see section 2), and therefore no suitably established procedures.

Finally, software publication in many cases is – or is perceived as – an arduous manual task that includes typing metadata into web forms provided by the target *publication repository*. Recently, technical solutions have been developed to automate this process. The GitHub-Zenodo integration[48] automates the publication of software developed in a publicly accessible GitHub *source code repository* whenever a release is created in the respective repository. When the repository contains a Citation File Format[49] or a Zenodo specific file, the metadata it contains is used to create the record on Zenodo.[50] While the latter approach only works for the combination of GitHub and Zenodo, the project HERMES[51] develops a concept[52] and software (hermes[53]) that can be automatically run in continuous integration systems. The software works with any combination of *source code repository* and *publication repository* for which an implementation in the hermes software exists. hermes is extensible via plugins; it harvests and processes software metadata from different sources in the *source code repository*, including metadata files, version control history, and platform APIs.

# 3 Software Citation

Software publication is a prerequisite for formal software citation as it provides a distinct set of metadata for software versions, uniquely identifiable through a persistent identifier. Software citation in turn makes good on the promise of software publication to improve reproducibility through persisted evidence of methods applied in research, and to enable credit for software authors.

## 3.1 The Principles of Software Citation

The principles of software citation[54] state that the software itself must be formally cited (i. e., like papers are cited). Citations should facilitate credit and attribution for authors and provide persistent unique identification of software. Citation should further facilitate access to the software itself, as well as to associated information that makes it reusable: persistent metadata, documentation, data, etc. And finally,

47 Eisty, Nasir U., Jeffrey C. Carver. "Developers Perception of Peer Code Review in Research Software Development." *Empirical Software Engineering* 27,1 (October 27, 2021): 13. doi:10.1007/s10664-021-10053-x; Petre, Marian, Greg Wilson. "Code Review For and By Scientists." *ArXiv* (2014). doi:10.48550/ARXIV.1407.5648.

48 Last checked 04.07.2023. http://web.archive.org/web/20230608105913/https://docs.github.com/en/repositories/archiving-a-github-repository/referencing-and-citing-content.

49 Druskat et al. 2021.

50 Last checked 04.07.2023. https://zenodo.org.

51 Last checked 04.07.2023. https://software-metadata.pub.

52 Druskat et al. 2022.

53 Meinel, Michael et al. "Hermes." *Python* (2022; repr., March 16, 2023). Last checked 04.07.2023. https://github.com/hermes-hmc/workflow.

54 Smith et al. 2019.

software citations should be as specific as necessary with regards to the exact version that was used[55] or is cited.

When the software citation principles are followed, software can become part of citation graphs of research output, with a resolution up to single versions.[56] This would allow a better understanding both of how research was conducted and how research results were obtained, which in turn would enable better reproducibility of these results and foster research software reuse. Additionally, bibliometric, authorship network and cooperation analyses and reporting should be extended to include software in line with the growing recognition and support for the "San Francisco Declaration of Research Assessment"[57] (DORA).

To enable formal software citation, the complete and correct citation metadata must be available for the software (version) that is cited. Ideally, these metadata can be resolved from a PID to a software publication. As a prerequisite for publication under a PID, the metadata must first be known, understood, and provided by the software project. For text outputs, there are normative criteria that define, for example, authorship.[58] For software, this is still work in progress.[59] Meanwhile, the software projects themselves are responsible for compiling and providing the correct and complete citation metadata for each version of their software. To support this effort, the Citation File Format (CFF[60]) provides a schema for human- and machine-readable citation metadata files for software. The format is supported by major platforms such as GitHub and Zenodo/InvenioRDM, which automate the processing of the citation metadata for human use, and for automated publication of software from the *source code repository*. The free and open source reference managers Zotero[61] and JabRef[62] support import of the metadata from CFF files.

If research software is treated on par with traditional outputs such as books or papers, this also means that software should cite its own references, for example the software packages it reuses and depends on. CFF supports the provision of references for software. If software cites other software in this way, this enriches research citation graphs further, so that it becomes possible to trace dependencies across software projects, identify critical projects for research domains or research as a whole, and, finally, to calculate transitive credit[63] for software projects and versions that are not used by researchers directly.

## 3.2 Challenges in Software Citation

The practice of software citation is still insufficient, with software being "mentioned" rather than formally cited in the majority of research reporting. These mentions range from citing a paper describing the software instead of citing the software itself, citations of project websites, URLs in footnotes, instrument-like in-text mentions and in-text mentions of just the software name to not even mentioning the name of the software[64] (see also Schindler et al.[65]). This situation is likely to change, as an increasing number of publishers define software citation policies.[66]

Similarly, correct and complete citation metadata is not yet provided by default in research software projects. For compiling the authorship part of such metadata, it is not yet defined what the criteria are for software authorship (see section 3). Where they are provided, the form in which they are provided does not always follow machine-readable standard formats such as the Citation File Format[67] or R CITATION files.

An issue for tracking software citations, and therefore for evaluating software work based on citations, is the lack of representation of PIDs for software in some of the most-used indexing services, e. g., Google Scholar[68] and databases (Web of Science, Scopus, Dimensions). At the time of writing,

55 User-facing software should be cited by users, while for its dependencies, the cited software (publication) should provide citation metadata, similar to cited publications in a references list of a paper. In source code repositories, this can be done with metadata files in the Citation File Format (Druskat et al. 2021).

56 Druskat, Stephan. "Software and Dependencies in Research Citation Graphs." *Computing in Science & Engineering* 22,2 (March 2020): 8–21. doi:10.1109/MCSE.2019.2952840.

57 Last checked 04.07.2023. https://sfdora.org.

58 International Committee of Medical Journal Editors. "ICMJE Recommendations: Defining the Role of Authors and Contributors." Last checked 10.05.2019. http://web.archive.org/web/20190905080833/http://www.icmje.org/recommendations/browse/roles-and-responsibilities/defining-the-role-of-authors-and-contributors.html#two.

59 Leem, Deborah et al. "SORTÆD: Software Role Taxonomy and Authorship Definition." *Zenodo* (May 4, 2023). doi:10.5281/zenodo.7896456.

60 Druskat et al. 2021.

61 Last checked 04.07.2023. https://www.zotero.org/.

62 Last checked 04.07.2023. https://www.jabref.org/.

63 Katz, Daniel. "Transitive Credit as a Means to Address Social and Technological Concerns Stemming from Citation and Attribution of Digital Products." *Journal of Open Research Software* 2,1 (July 9, 2014): e20. doi:10.5334/jors.be.

64 Druskat et al. *forthcoming*; Howison, Bullard 2014.

65 Schindler, David et al. "The Role of Software in Science: A Knowledge Graph-Based Analysis of Software Mentions in PubMed Central." *PeerJ Computer Science* 8 (January 14, 2022): e835. doi:10.7717/peerj-cs.835.

66 See https://www.chorusaccess.org/resources/software-citation-policies-index/. Last checked 04.07.2023.

67 Druskat et al. 2021.

68 Last checked 04.07.2023. https://scholar.google.com.

the only notable example is DataCite Commons,[69] a search service for DataCite DOIs that can be queried for software and link citation metadata.

There are additionally some fundamental challenges,[70] for example involving the different modes software can take: as a concept or project, and as a version; as open, closed, or *inner source software*; as published or unpublished software. This leads to issues with the correct identification of software for citation: how can the unpublished version 14.3 of the closed source SAS/STAT software[71] be formally cited?

Further practical issues relate to the correct mapping of software metadata to metadata formats that do not provide dedicated fields for software, or the lack of citation styles that cater for software[72] and are supported by publishers.

At the root of all these challenges is the need for a change in culture and practice of how different stakeholders in research accommodate research software and its citation. Therefore, the interaction with and between these stakeholders in order to bring software citation into scholarly culture is central and should include libraries, disciplinary communities, publishers, repositories and registries, indexers, research funders, and academic institutions.[73] As an example for a change in stakeholder policy for research software, the Helmholtz Association of German Research Centres has implemented an indicator to track citable software publications.[74] This indicator can potentially be used in research evaluation to include and evaluate software work.

# 4 What can Libraries do to Become "Research Software-ready"?

The acknowledgment of research software as an indispensable part of research across disciplines, and a research output in its own right, is increasing.[75] Academic libraries can support this through a variety of measures that align with their core mission and start to recognize research software as something that is in their remit.

In the following, we identify areas of activity for libraries to support research software.

## 4.1 Infrastructure

Academic libraries often provide technical infrastructure through which their services are made accessible to their users. Infrastructure is used in partnership with researchers over the life cycle of research projects, e. g., to help them fulfill mandates for archiving research outputs. As these mandates shift to include research software alongside research data (see the Code of Conduct for Good Research Practice by Deutsche Forschungsgemeinschaft[76]), libraries and their partners (e. g., institutional IT departments or computing centers) are required to adapt their technical infrastructure to support the new mandates.

Infrastructure to support the publication and citation of research software may include *publication repositories* or *software registries*. Both types of system must be able to ingest, and correctly and distinctively represent software and software metadata. In particular, *publication repositories* have often been designed with research data in mind, and must be evaluated/adapted for use with research software. This is true especially with regard to their metadata models, as software must be described with other metadata than data, e. g., in the context of citation.[77] Relevant software metadata such as runtime requirements, dependencies, programming language, operating system or development status do not apply for data at all. Other metadata seem the same semantically, whereas their values may be valid for one but not the other: license (software licenses do not apply to data); legal status (unlike data, software is creative work); different contribution roles; different versioning schemes, etc. The CodeMeta[78] community standard is one example for such a descriptive software metadata schema, along with multiple other options using linked data and knowledge graphs (see Druskat et al.[79]). In summary, *publication repositories* must be research

**69** Last checked 04.07.2023. https://commons.datacite.org/.

**70** Katz, Daniel S. et al. "Software Citation Implementation Challenges." *ArXiv* (May 21, 2019). http://arxiv.org/abs/1905.08674.

**71** Last checked 04.07.2023. https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.3/statug/titlepage.htm.

**72** A notable exception here is the biblatex-software package (Di Cosmo, Roberto. "biblatex-software. Version 1.2-1." LaTeX, June 1, 2020. https://ctan.org/pkg/biblatex-software).

**73** Katz et al. 2019.

**74** Helmholtz-Gemeinschaft. "Helmholtz Open Science Policy. Version 1.0. Approved in the 119th General Assembly of the Helmholtz Association on 20–21 September 2022." (2022). doi:10.48440/os.helmholtz.056.

**75** Jay, Haines, Katz 2021; Doerr et al. 2019.

**76** Deutsche Forschungsgemeinschaft. "Guidelines for Safeguarding Good Research Practice. Code of Conduct." *Zenodo* (April 20, 2022). doi:10.5281/zenodo.6472827.

**77** Katz, Daniel S. et al. "Software vs. Data in the Context of Citation." *PeerJ Inc.* (December 2016). doi:10.7287/peerj.preprints.2630v1.

**78** Jones et al. 2017.

**79** Druskat et al. 2022: 6.

software-ready themselves if they are to be employed by libraries to become research software-ready.

As some publishers have established data and software management policies detailing how and where software (and data) related to a paper publication should be made available,[80] libraries or their infrastructure partners should ensure that accepted deposits can reference related software.

It is important to note that libraries are not required to provide and administrate all required technical infrastructure themselves. They may just as well make use of existing infrastructure provided elsewhere. A good example of this would be the reuse of metadata from an existing general purpose *publication repository* such as Zenodo in a *software registry* provided by a library, where the registry includes only the metadata for software that has been produced with participation from members of the library's institution. This is the case with the instance of Research Software Directory[81] run by the Helmholtz Association of German Research Centers at https://helmholtz.software. The catalogue collects software metadata from *source code repositories* as well as from Zenodo, and presents them in a way that makes the software more findable (through search engine optimization), accessible (through the presentation of relevant links to archived versions), and reusable (through descriptions and other metadata).

Regardless of what infrastructure is provided, interoperability with other systems should be assured and documented, so that metadata can be exchanged and deposited, via directories such as re3data.org, and through standards such as OAI-ORE, SWORD, and OAI-PMH. Furthermore, the operation and administration of infrastructure services should follow community standards, e. g., Garijo et al.,[82] for *software registries* and *publication repositories*.

## 4.2 Training and Support

Academic libraries commonly conduct training courses for students and researchers, teaching them how to use the library's services, including searching for research outputs and how to cite outputs used in research. Here, the curricula should be extended to cover finding, retrieving, and working with research software, the principles of software citations based on Smith et al.[83], and working with reference managers that support software types. When educating users about further topics that have relevance for research software, the contents provided by the library should include the software-relevant information, e. g., with regard to intellectual property, applicable policies and good scientific practice, licensing, etc.

Where libraries offer advanced training and support, e. g., around the research life cycle and research data management, these offers should also be extended to cover research software. This can include, for example:

- helping users determine suitable software publication practices for their project, with regard to the place of publication, the metadata that goes with the publication, how to prepare software for publication, and how to publish software in practice (see Druskat et al.[84]);
- helping users make their software citable by pointing them to relevant standard formats[85] and tools[86],[87].

## 4.3 Software Management and Curation

One of the core competencies of libraries is the curation and management of research outputs and their metadata. Traditionally, these have mainly been textual outputs, until research data management (RDM) was put into the scope of librarianship decades ago,[88] and has grown into an important task for the academic library.

Libraries can support research software in this field by transferring and applying their expert knowledge in research data management and relevant infrastructure to software. In advanced cases, libraries can curate research software publications as digital objects including software artifacts, and can help arrange for their long-term

**80** See https://www.chorusaccess.org/resources/chorus-for-publishers/publisher-data-availability-policies-index/. Last checked 04.07.2023.

**81** Cahen, Ewan Jacov et al. "Research Software Directory (as a Service)." *GitHub* (May 2023). https://github.com/research-software-directory/RSD-as-a-service. Last checked 04.07.2023.

**82** Garijo, Daniel et al. "Nine Best Practices for Research Software Registries and Repositories." *PeerJ Computer Science* 8 (August 8, 2022): e1023. doi:10.7717/peerj-cs.1023.

**83** Smith et al. 2019.

**84** Druskat et al. 2022.

**85** Druskat et al. 2021.

**86** For example, cffinit, a webform to create files in the Citation File Format, available at https://citation-file-format.github.io/cff-initializer-javascript/. Last checked 04.07.2023.

**87** Spaaks, Jurriaan H. et al. "Cffinit." *Zenodo* (January 17, 2023). doi:10.5281/zenodo.7543718.

**88** Cook, Michael N. *Numeric Data Products and Services: A SPEC Kit.* SPEC Kit,0160-3582;263. Washington, D.C.: Association of Research Libraries, Office of Leadership and Management Services, 2001.

availability in an archive, e. g., Software Heritage[89]. In any case, libraries can safeguard the quality of metadata for software outputs through curation – with a particular focus on citability – as they advance their knowledge base with research software specifics. These activities are independent of whether the library provides relevant infrastructure itself or not. Additionally, libraries can develop or adapt, implement and mandate software management plans (SMPs) in analogy to data management plans[90].

### 4.4 Policies

Libraries are important stakeholders, and valuable implementation partners, for policies for research software in the context of software publication and software citation, but also academic evaluation based on research software work (e. g. at Helmholtz-Gemeinschaft[91]). They can provide expert knowledge and infrastructure to assess and evaluate research software outputs through scientometric information systems, *publication repositories*, and *software registries*.

They should also be (or get) involved in the shaping of policies that impact research outputs such as software. This includes current and future challenges, such as handling mixed-content deposits and publishing workflows in a manner that enables reproducibility.[92]

Additionally, they can help build interfaces to related communities, such as the cultural heritage GLAM community, to improve knowledge exchange (see Sufi et al.[93]; Bouquin et al.[94]).

## 5 Conclusion

Academic libraries have a long history as caretakers for research outputs and helping users find, use, and cite them. Research data has been acknowledged as important research artifacts that deserve the attention of and treatment by librarians. Today, as research software is increasingly acknowledged as valid research output in its own right, libraries should broaden their focus and understand research software as a type of research output they manage. In practice, they can help users deal with software according to good practice and raise awareness for the specific requirements of research software in general.

In this paper, we introduced two important aspects of what libraries' engagement with research software should focus and build on: software publication and software citation. We introduce the current understanding of software publication as uniquely identifiable persistent metadata and artifact deposition in *publication repositories*, software citation as an adaptation of traditional citation practice to meet the specific needs of software, and the intersection of both practices and their significance for academic libraries.

In order to become research software-ready, libraries can engage with research software in a number of areas of activity. They can provide and reuse technical infrastructure that caters to software as a research output and its documentation in a library context. This includes mainly *publication repositories* and/or *software registries* that can ingest and represent research software artifacts and their specific metadata.

Libraries can also extend their education and training of researchers and other library users to include software publication and software citation in theory and practice, and their research data management and curation services and support to include research software. The latter may also include the development and support of software management plans.

Finally, libraries can engage with other stakeholders in research software to create and implement policies to further research software sustainability and the successful adoption of the FAIR Principles for Research Software.

**Acknowledgments:** We would like to thank Dr. Tom Honeyman (Software Program Manager, Australian Research Data Commons) and Dr. Daina Bouquin (former Head Librarian, John G. Wolbach Library at Harvard Smithsonian Center for Astrophysics, now Data Operations and Research Manager at the US National Parks Conservation Association) for their immensely helpful input in the early stages of writing this

**89** Di Cosmo, Roberto, Stefano Zacchiroli. "Software Heritage: Why and How to Preserve Software Source Code." *IPRES 2017 – 14th International Conference on Digital Preservation* (Kyoto, Japan, 2017): 1–10. https://hal.archives-ouvertes.fr/hal-01590958. Last checked 04.07.2023.

**90** Martinez-Ortiz, Carlos et al. "Practical Guide to Software Management Plans." *Zenodo* (January 31, 2023). doi:10.5281/zenodo.7589725.

**91** Helmholtz-Gemeinschaft. 2022.

**92** Soiland-Reyes et al. 2021.

**93** Sufi, Shoaib et al. "Report on the Workshop on Sustainable Software Sustainability 2019 (WOSSS19)." *Zenodo* (July 8, 2020). doi:10.5281/zenodo.3922155.

**94** Bouquin, Daina et al. "Advancing Software Citation Implementation (Software Citation Workshop 2022)." *arXiv* (February 15, 2023). doi:10.48550/arXiv.2302.07500.

paper. SD and OB acknowledge funding from the Initiative and Networking Fund of the Helmholtz Association via Helmholtz Metadata Collaboration project HERMES (*ZT-I-PF-3-006*). AS acknowledges the support of the Cluster of Excellence "Matters of Activity. Image Space Material" funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2025 – 390648296.

## Glossary

***inner source software*** The application of open source principles such as open communication, openness to third-party contributions, open source governance models, and self-management to software projects exclusively within an organizational unit, e.g., an academic institution.

***publication repository*** An archive that hosts digital artifacts and standardized metadata for the artifacts. Each artifact is addressable with a unique identifier. Publication repositories are usually instances of repository platforms such as Dataverse,[95] InvenioRDM[96] or DSpace,[97] and may be run internally by institutions, or accessibly to the general public. A well-known example of the latter is Zenodo,[98] an open access general purpose repository run by CERN.

***software registry*** An index or catalogue of software metadata. In contrast to a publication repository, a registry does not store software artifacts, only information about them, to aid their discovery.[99] Other terms – currently emerging in the research software infrastructure community – for software registry are software catalogue and software directory.

***source code repository*** A repository, often a version control system repository, that hosts software source code and other files, such as documentation and metadata files. Source code repositories may be hosted publicly on a software development platform such as GitHub, GitLab, or similar.

## Author information

**Stephan Druskat**
Institute for Software Technology
German Aerospace Center (DLR)
Rutherfordstr. 2
12489 Berlin
**stephan.druskat@dlr.de**
https://orcid.org/0000-0003-4925-7248

**Oliver Bertuch**
Forschungszentrum Jülich GmbH, Central Library
Jülich
**o.bertuch@fz-juelich.de**
https://orcid.org/0000-0002-2702-3419

**Alexander Struck**
Cluster of Excellence Matters of Activity
Humboldt-Universität zu Berlin
**Alexander.Struck@hu-berlin.de**
https://orcid.org/0000-0002-1173-9228

95 King, Gary. "An Introduction to the Dataverse Network as an Infrastructure for Data Sharing." Sociological Methods & Research 36,2 (November 1, 2007) doi10.1177/0049124107306660.
96 Last checked 04.07.2023. https//inveniosoftware.org/products/rdm/.
97 Last checked 04.07.2023. https://dspace.lyrasis.org/.
98 European Organization For Nuclear Research and OpenAIRE. "Zenodo: Research. Shared." (2013). doi:10.25495/7GXK-RD71.
99 Garijo et al. 2022.