FAIRSFAIR
Fostering Fair Data Practices in Europe

Project Title	Fostering FAIR Data Practices in Europe
Project Acronym	FAIRsFAIR
Grant Agreement No	831558
Instrument	H2020-INFRAEOSC-2018-4
Topic	INFRAEOSC-05-2018-2019 Support to the EOSC Governance
Start Date of Project	1st March 2019
Duration of Project	36 months
Project Website	www.fairsfair.eu

M2.15 Assessment report on 'FAIRness of software'

Work Package	WP2- FAIR practices: semantics, interoperability and services
Lead Author (Org)	Morane Gruenpeter (INRIA)
Contributing Author(s) (Org)	Roberto Di Cosmo (INRIA), Hylke Koers (SURF), Patricia Herterich (DCC), Rob Hooft (DTL), Jessica Parland-von Essen (CSC), Jonas Tana (CSC), Tero Aalto (CSC), Sarah Jones (DCC)
Due Date	30.09.2020
Date	16.10.2020
Version	1.1
DOI	10.5281/zenodo.4095092

Dissemination Level

X	PU: Public
	PP: Restricted to other programme participants (including the Commission)
	RE: Restricted to a group specified by the consortium (including the Commission)
	CO: Confidential, only for members of the consortium (including the Commission)

Abstract

Software has an important place in academia and as such it has an important place in the FAIR ecosystem. Software can be used throughout the research process; however it can also be an outcome of the research process. Distinguishing between these different roles is essential for any assessment of the `FAIRness of software'.

This is the first milestone of the FAIRsFAIR project focused specifically on software as a digital object. In this report we discuss the state-of-the-art of software in the scholarly ecosystem in general and in the FAIR literature in particular. We identify the challenges of different stakeholders when it comes to finding and reusing software. Furthermore, we present an analysis of nine resources that call for the recognition of software in academia and that present guidelines or recommendations to improve its status - either by becoming more FAIR or by improving the curation of software in general. With this analysis we demonstrate to what extent each of the FAIR principles is seen as relevant, achievable and measurable; and in what sense it benefits software artifacts. Finally, we present 10 high-level recommendations for organizations that seek to define FAIR principles or other requirements for research software in the scholarly domain.

Versioning and contribution history

Version	Date	Authors	Notes
0.3	21.7.2020	Morane Gruenpeter and all contributing authors	Main structure and content description
0.4	18.8.2020	Morane Gruenpeter and all contributing authors	First draft for T2.4 review- methodology for the analysis of the literature
0.5	1.9.2020	Morane Gruenpeter and all contributing authors	
0.6	9.9.2020	Morane Gruenpeter and all contributing authors	Software in the FAIR ecosystem specified and FAIR analysis completed
0.7	15.9.2020	Morane Gruenpeter and all contributing authors	Updated introduction Complete summary of finding and recommendations rational
0.8	24.9.2020	Morane Gruenpeter and all contributing authors	Added full analysis table Updated recommendations Draft for internal review
0.9	29.9.2020	Morane Gruenpeter and all contributing authors	Draft for PCO

1.0	30.9.2020	Morane Gruenpeter and all contributing authors	Shared 1.0 version on FAIRsFAIR archive
1.1	16.10.2020	Morane Gruenpeter, all contributing authors and experts from the FAIR4RS	Published first version on Zenodo for community review

Disclaimer

FAIRsFAIR has received funding from the European Commission's Horizon 2020 research and innovation programme under the Grant Agreement no. 831558. The content of this document does not represent the opinion of the European Commission, and the European Commission is not responsible for any use that might be made of such content.

Abbreviations and Acronyms

ARDC	Archival, Reference, Description and Credit
CURE	Curating for Reproducibility
EOSC	European Open Science Cloud
FAIR	Findable, Accessible, Interoperable, Reusable
FAIR4RS WG	FAIR for Research Software Working Group
FOSS	Free and Open Source Software
FSFE	Free Software Foundation Europe
PID	Persistent Identifier
RDA	Research Data Alliance
ReSA	Research Software Alliance
SCID WG	Software Source Code Identification Working Group
SCI WG	Software Citation Implementation Working Group
SIRS TF	Scholarly Infrastructures for Research Software Task Force
SPDX	Software Package Data Exchange
SSC IG	Software Source Code Interest Group
T2.4	Task 2.4 in Work Package 2: FAIR services and software
WP2	Work Package 2 in the FAIRsFAIR project: FAIR practices: semantics, interoperability and services

FAIRSFAIR
Fostering Fair Data Practices in Europe

Table of contents

1. Introduction **6**

2. FAIR principles for software at the start of 2020 **7**

2.1 Software as part of a FAIR ecosystem 7

2.1.1 Related initiatives 9

2.1.2 Software in FAIRsFAIR 12

2.2 Do software artifacts require different FAIR principles? 12

3. Challenges **16**

3.1. Software dependencies and environment - technical challenge 16

3.2 Documentation 17

3.3 Accessibility & Licensing 17

3.4 Time and skill 17

3.5 Quality control 17

3.6 Software sustainability & management plan 18

3.7 Metadata 18

4. FAIR analysis of research software guidelines **19**

4.1 Methodology 20

4.2 Compendium of FAIR software analysis 22

4.3 Other insights and recommendations from the literature 22

4.4 Summary of Findings 24

5. Towards FAIRness of software **26**

5.1 Existing mechanisms and components for software 26

5.1.1 Software identification 26

5.1.2 Software metadata and vocabularies 27

5.1.3 Software licenses and SPDX 28

5.1.4 Software Curation 29

4.1.5 Software artifact evaluation and badging 30

5.2 The landscape of existing infrastructures 30

5.2.1 Software archives and institutional repositories 31

5.2.2 Software journals and publishers 31

5.2.3 Software registries / indexers / aggregators 32

5.2.4 Research software training 32

6. Recommendations **32**

7. Conclusion **35**

8. Acknowledgements **36**

Annex A: FAIRsFAIR Task 2.4 Statement of Work **37**

Annex B: Complete analysis of software guidelines **40**
B.1 Findable 40
B.2 Accessible 44
B.3 Interoperable 48
B.4 Reusable 51

Annex C: Infrastructures and existing implementations catering software **55**
C.1 Software archives and institutional repositories 55
C.1.1 Software Heritage archive 55
C.1.2 Zenodo 56
C.2 Software journals and publishers 57
C.2.1 SoftwareX 57
C.2.2 Journal of Open Source Software (JOSS) 58
C.3 Software registries / indexers /aggregators 59
C.3.1 swMath 59
C.4 Research software training 60
C.4.1 The Carpentries 60

Bibliography **61**

1. Introduction

This `**FAIRness of software**' assessment report is the second milestone report produced by the FAIR services and software task (T2.4), included in the FAIR practices work package (WP2) under the FAIRsFAIR[1] European project. The FAIRsFAIR project's goal is to propose solutions, standards and recommendations to implement the FAIR principles in the research object's life cycle. Surveying the landscape of FAIR activities to create a basis for harmonization, while identifying overlaps, divergences and challenges is a fundamental first step before proposing solutions. Early on, T2.4 established that services and software should be assessed separately in such an endeavour, resulting in the addition of this milestone to ensure all T2.4 objectives are addressed. The choice to assess services and software on different tracks reflects the distinction when it comes to software and its triple role in academia, as stated in (Clément-Fontaine, 2019): "Software can be a tool, similarly to services, but it has an important role as a research result and as a research object".

Software is essential in research for each role that it plays. Furthermore, source code has particular importance because of the scientific knowledge and the logic of data transformation embodied in the source code (Di Cosmo and Zacchiroli, 2017). With this in mind, different stakeholders and initiatives in research have recently articulated the need to better understand software and to include source code as a separately recognized digital object in a `FAIR ecosystem' (European commision, 2018). In this report we review the state-of-the-art view of software in a `FAIR ecosystem', through an analysis of existing literature about the FAIR principles applied to software and guidelines or recommendations on related subjects, including software citation, software curation and the place of software in academia. We also provide high-level recommendations, using an analytical approach, to help evaluate the adequacy of each principle with regards to software and especially to source code.

The report is organized as follows: Section 2 is an overview of the FAIR ecosystem and related initiatives in the academic domain, trying to answer the question of whether software artifacts require a different set of FAIR principles . Section 3 is concerned with challenges in finding and re-using software as identified by the community, which helped us identify gaps between the existing literature and current technical and sociological challenges. Section 4 then reviews and analyzes the existing literature on addressing the challenges in applying the FAIR principles to research software. Section 5 offers a panorama of infrastructures and services providing support for research software to become (increasingly) FAIR and outlining benefits and limitations for each solution. Finally, we end this report with a summary of findings and recommendations to support initiatives such as the Research Data Alliance / Research Software

[1] https://www.fairsfair.eu

Alliance / FORCE11 FAIR for research software working group (FAIR4RS[2]), which was launched in June 2020 with the intention of creating a community definition of the FAIR principles for research software.

2. FAIR principles for software at the start of 2020

The FAIR guiding principles (Wilkinson et al. 2016) identified the difficulties of discovering and reusing data, and called for infrastructures to enhance the machine-actionability of their services. This first publication, which was specifically targeted to data, also states that FAIRness should be reached for all research objects including algorithms, tools and workflows. Since the publication of the FAIR principles, different academics and working groups have published articles suggesting that the FAIR principles as written do not naively apply to software, and some adjustments and expansions are needed when assessing the FAIRness of software.

We describe the landscape of such efforts, here divided into two main sections:

- A top-down view of the FAIR ecosystem and the role that software plays in it, alongside a description of the related initiatives engaging in FAIR software activities;
- A review of the current state of FAIR assessment for software, focusing on software as a research outcome and as the object of research. Software that is used in the academic process as a tool might be considered a service, which is detailed in the FAIRsFAIR assessment report on FAIRness of services (Koers et al., 2020). In this case, the software can be considered `FAIR enabling` checking if the service or tool is helping the data it acts upon to become more FAIR, and not directly as a FAIR object. Note that software that was produced as a research outcome can also be a research tool in a different setting, for a different team for example.

2.1 Software as part of a FAIR ecosystem

A FAIR ecosystem, as described in the report *Turning FAIR into reality* (European commision, 2018), is a highly distributed ecosystem requiring technical mechanisms linking resources, and social mechanisms to define specifications, standards and protocols. Both the FAIR guiding principles and the report *Turning FAIR into reality* state that software is one type of digital object to which the principles should apply. Nonetheless, software can be found at different levels:

- Infrastructure components (registries, repositories, etc.) and protocols are implemented with software;
- services are software instances that can be deployed on online platforms;

[2] https://www.rd-alliance.org/groups/fair-4-research-software-fair4rs-wg

- during the research process, software can be used as a tool, with or without modifications;
- the research process can yield a software outcome;
- software source code can play the same role as a dataset that can be observed, analyzed and re-used.

In this report we focus on software representing a research outcome.

In the FAIRsFAIR document "FAIR Ecosystem Components: Vision" (L'Hours & Von Stein, 2020) a diagram for the ecosystem components is presented to better understand the relations between infrastructure, objects and actors. Figure 1 shows a modified version of the FAIR ecosystem diagram with the added software icon (</>). We see that software, in the purple rectangle, has a place of its own in the ecosystem. However, software has a place in multiple layers, depending on the purpose it serves. To illustrate this idea, we added the software icon where software may be relevant.

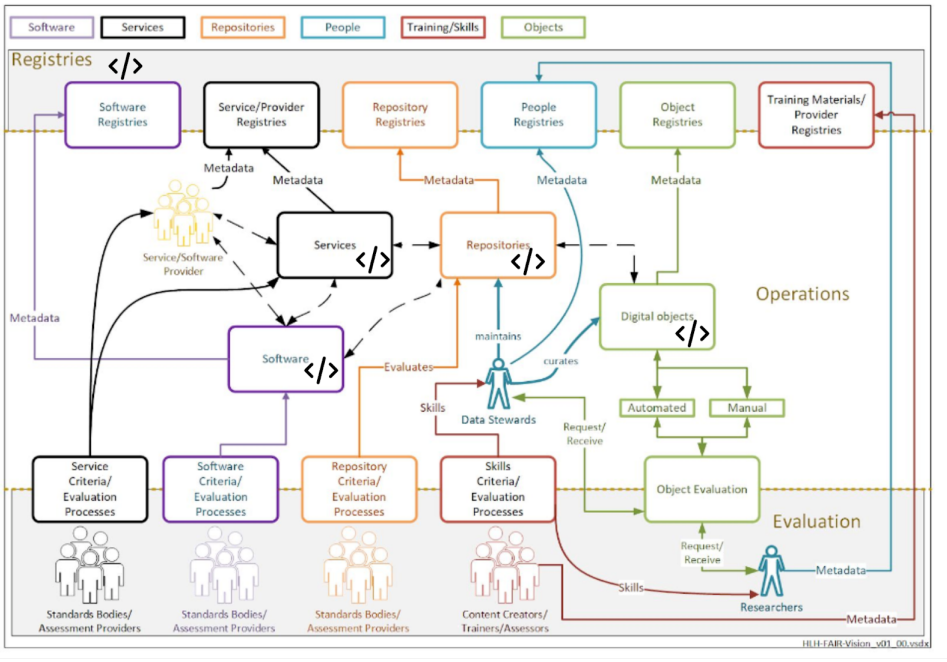

Figure 1: FAIR vision: Ecosystem components, to highlight the software roles in the Ecosystem, the symbol </> was added (Original diagram 3 from L'Hours & Von Stein, 2020)

Software is all around us, with a majority of it developed outside of academia in industry and developer communities (RDA/FORCE11 Software Source Code Identification WG, 2020). In this report we address software both developed and used in academia. We do not attempt to define research software, instead use resources where research software is defined and where

the FAIR principles are interpreted specifically for research software. Furthermore, software is a complex object and different aspects must be taken into account; its usage, size, life span, structure, authorship, community around it, and the authority over the software (Alliez et al., 2019). We propose to keep these aspects in mind when reviewing software projects and assessing software artifacts as FAIR digital objects.

2.1.1 Related initiatives

Many different initiatives have undertaken the challenge of making software a first class citizen in the scholarly ecosystem. These initiatives' main objectives are to establish recognition of software as a research output and to improve research reproducibility which can include software. One of the themes coming back in these initiatives is the question of the applicability of the FAIR principles to software. In the following paragraphs we will detail these initiatives with their background and goals:

- The Research Data Alliance (RDA)[3] Software Source Code IG (SSC IG) was founded in 2017, after a Birds of a Feather (BoF) session at the 9th RDA Plenary in spring of that year. It discusses issues of identification, management, sharing, discovery, archiving and provenance of software source code, reviewing and revising metadata for describing and discovering source code, developing guidelines for managing, describing and publishing software source code, collecting and publishing use cases of current examples and practices, and contributing software related expertise to other groups in the RDA which have a software aspect.
  In the SSC IG sessions, two RDA working groups were discussed and created: the Software source Code Identification Working Group [SCID] WG (also a FORCE11[4] WG) and the FAIR for research software [FAIR4RS] WG (also a working group of FORCE11 and a collaboration with ReSA[5]).
- The SCID was launched in March 2019 to collect and discuss use cases involving software source code identification with the intention of understanding and capturing the landscape and coming to a consensus, since many identifier schemes for software, including both intrinsic and extrinsic identifiers, exist. The WG output was published for community review in July 2020. The WG output[6] is a collection of use cases and identifier schemes that are relevant for software source code identification as well as a summary analyzing these findings. This was a necessary first step before making any

[3] Research Data Alliance (RDA): https://www.rd-alliance.org/
[4] FORCE11 : https://www.force11.org/
[5] Research Software Alliance (ReSA): https://www.researchsoft.org/
[6] https://doi.org/10.15497/RDA00053

recommendations that might be only applied in the scholarly ecosystem and might be detached from the case of software in industry.

- FAIR4RS was launched at the end of June 2020 and aims to define the FAIR principles for research software with community support. The working group will produce guidelines on how to apply the FAIR principles for research software (based on existing frameworks) and adoption examples. This is the seminal work in the area, with initiatives such as the EOSC FAIR WG recognising this as the community forum for taking forward the FAIR principles for software, services and workflows.[7]

  During the summer of 2020, the working group formed four different subgroups with the following goals:

  - ❖ Subgroup 1: "A fresh look at FAIR for Research Software" will examine the FAIR principles in the context of research software from scratch, not based on pre-existing work.
  - ❖ Subgroup 2: “FAIR work in other contexts” will examine efforts to apply FAIR principles to different forms including workflows, notebooks and training material, to provide insights for the definition and implementation of FAIR principles for research software.
  - ❖ Subgroup 3: “Definition of research software” will review existing definitions of research software and will specify the scope for the WG outputs.
  - ❖ Subgroup 4: “Review of new research related to FAIR Software” will review new research around FAIR software that has come out since the release of the Towards FAIR principles for research software paper 10.5281/zenodo.3904139 (Lamprecht et al., 2019) in August 2019.

- The CURE & FAIR working group[8] is a new RDA working group that was created after a BoF session at P14: Curating for FAIR and Reproducible Data and Code. The abbreviation CURE stands for Curating for Reproducibility[9]. The working group's goal is to establish guidelines and standards to promote curated, reproducible and FAIR data and code.

- The Software Citation Implementation Working Group[10] (SCIWG) is a FORCE11 initiative, building on the Software Citation Working Group, which published the software citation principles (Smith et al. 2016). It started in 2017 with the following goals:
  - Endorse the software citation principles

[7] 10.5281/zenodo.3904139
[8] https://www.rd-alliance.org/groups/cure-fair-wg
[9] https://osf.io/f4jtb/
[10] https://www.force11.org/group/software-citation-implementation-working-group

- Propose guidelines on how to implement the principles (depending on the stakeholder)
- Promote the implementation of the principles
- Test implementations of the principles.

Citation and FAIR are closely related; FAIR objects with rich metadata and persistent identifiers provide all elements needed for a citation and the incentive of being cited might encourage researchers to spend effort on making their outputs FAIR. But there are key differences too, as citations are also meant to provide credit to the authors of software, which is a creation of human ingenuity, and falls under copyright law, unlike most datasets. Hence, proper attribution of research software, which falls squarely out of the core objectives of the FAIR movement, is of paramount importance. World class research institutions in Computer Science have been handling these issues internally for decades, and recently efforts have been made to share widely the lessons learned, for example by the software citation working group at Inria (Alliez et al. 2019).

Considerable effort has been spent over the past few years to discuss the many complex issues related to software citation - that involve proper identification, description and attribution of software artifacts - but while there is agreement about the importance of software citation in general, there is not yet a formal, globally accepted standard on how software should be cited. Difficulties and challenges of software citation are described in the SCIWG output: Software Citation Implementation Challenges (Katz et al., 2019). As an example of how far this issue is from being solved, we remark that only in May 2020 a bibliographic style for software artifact, `biblatex-software`[11] was made available for users of the `biblatex` bibliography processing tool included in the popular LaTeX system for research articles (Di Cosmo, 2020), and is only now starting to be included in standard publication formats (see for example the *Journal of Theoretical, Computational and Applied Mechanics* https://jtcam.episciences.org/page/for-authors).

- In June 2020, the EOSC Architecture Working Group launched a task force on Scholarly Infrastructures for Research Software (SIRS), with the goal to survey the existing practices of infrastructure that deal with research software, from archives to publishers to catalogs, and make recommendations for building a lean architecture of infrastructures supporting Open Science. Interestingly, this work leverages an alternative approach for looking at research software, based on the identification of the functionalities that an infrastructure should provide, namely Archival, Reference, Description and Credit, or ARDC.

[11] https://www.ctan.org/tex-archive/macros/latex/contrib/biblatex-contrib/biblatex-software

2.1.2 Software in FAIRsFAIR

The work being undertaken in FAIRsFAIR Task 2.4 aims to capture the panorama of existing frameworks and documents addressing specifically software and evaluate the applicability of the FAIR principles to software, or related concepts to software.

Software has an evident place in the research lifecycle as a tool, but more importantly it can be a conclusive result of the research process. Software is part of the academic domain in many different disciplines. As such, it is fundamental to consider its curation and preservation by applying a set of guidelines to insure quality curation.
The FAIR principles are a good candidate to apply to software artifacts as they are also digital objects, however it is still unclear to what extent the principles as written are suited to software. Different academics and initiatives have addressed this question from various angles, sometimes specifically comparing the FAIR principles and sometimes addressing other quality curation aspects.

Moreover, we aim to highlight the challenges that *research software outcomes*, *software tools* and *software objects of research* face when considered as part of a FAIR ecosystem. Recommendations and guidelines on how to use and implement these recommendations are needed to improve software recognition and comprehension in a FAIR ecosystem and more broadly in the academic community.

2.2 Do software artifacts require different FAIR principles?

Modern research relies on software, yet software is different from other research outputs due to its nature. Therefore, digital objects requirements, as defined in the *FAIR guiding principles*, might not apply to software. It is important to recognize software as complexe object whose nature must be taken into account when producing FAIR principles for research software.
Before diving into the specific case of software artifacts, we propose a simplified summary of the FAIR guiding principles as written for research data (Wilkinson et al., 2016) and presented in Figure 2.
A **Findable** digital object is an object identified with a globally unique and persistent identifier (PID), which is registered with a set of rich metadata in a registry. These metadata can be searched and thus the digital object can be found. The identifier is part of the metadata record.
An Accessible digital object and its metadata can be retrieved with a standard communication protocol, which is open and free. Authentication might be used when necessary. Metadata is always accessible (even if the data is not).

An **Interoperable** digital object has metadata in a formal, shared vocabulary that is also following the FAIR principles. It also includes references to other metadata of other digital objects, creating links between resources.

A **Reusable** digital object has metadata with a plurality of accurate attributes, it is published/released with a clear license for usage and associated with its provenance. Also the metadata and digital object meet community standards.

Figure 2: Illustration of ANDS resources which reflect or crosscut the FAIR principles. Image: ANDS[12]. CC: BY 4.0

The *Turning FAIR into reality* report recommends that the FAIR guiding principles should be applied broadly (European Commission, 2018), which includes software source code, workflows and combination of these. However, it is clear from the report's action 16.2 that specific tailoring of the principles is needed in order to fit other objects.

Rec. 16: Apply FAIR broadly
FAIR should be applied broadly to all objects (including metadata, identifiers, software and DMPs) that are essential to the practice of research, and should inform metrics relating directly to these objects.

Action 16.1: Policies must assert that the FAIR principles should be applied to research data, to metadata, to code, to DMPs and to other relevant digital objects, as well as to policies themselves. Stakeholders: Policymakers.

[12] https://www.ands.org.au/working-with-data/fairdata/training

Action 16.2: The FAIR data principles and this Action Plan must be **tailored for specific contexts** - in particular to the relevant research field - and the precise application nuanced, while respecting the objective of maximising data accessibility and reuse. Stakeholders: Research communities; Data service providers; Policymakers

In June 2020, the EOSC's 'FAIR in practice' task force released `Six Recommendations for Implementation of FAIR Practice` (FAIR Practice TF, 2020) including a full section covering the FAIR practices for digital objects that are not research data (e.g software, services, tools and executable notebooks). One of the recommendations emphasises the need for translating the FAIR principles for different types of digital objects :

Recommendation n°5 : Recognise that FAIR guidelines will require translation for other digital objects and support such efforts.

Also the French national Committee for Open Science's Free Software and Open Source Project Group has published in 2019 the Opportunity Note *Encouraging a wider usage of software derived from research* (Clément-Fontaine, 2019):

*Recommendation n° 2 : Make sure **the specific nature of software** is recognized and not considered as "just data" particularly in the context of discussion about the notion of FAIR data.*

Following these recommendations, a FAIR ecosystem requires special attention to software. Software is different from data as exposed in (Katz et al., 2016), which in turns may require a different FAIRification process[13] and a different way to evaluate the FAIRness of software. Moreover, the FAIRness of software depends on the role it plays in the research life cycle.
Software that is used **as a tool** and is not a research product might be considered similarly to a service that serves the data or the research workflow. In that way this software can enable, respect or reduce the FAIR principles for other digital objects.
Software that was **created** during the research life cycle **as part of a research effort** is a research outcome and should have the same status as a research result (e.g., articles, theses, books, reports, datasets, etc.) It should be citable and its authors should receive proper credit.
Lastly, software can be an **object that is the subject of research**. In this case, software is similar to (source) data, noting that data can be both created during the research process and taken as an input. An example for this role would be the The Mining Software Repositories[14] (MSR) and

[13] https://www.go-fair.org/fair-principles/fairification-process/
[14] http://www.msrconf.org/

The International Conference of Software Engineering[15] (ICSE) conferences, for which large sets of source code are analyzed for the purpose of research.

With this distinction in mind, FAIR principles for software should be evaluated differently for each of the three different software facets :

- As a tool,
- As a research outcome or result; and
- As the object of research.

Trying to address all three at once can impose unnecessary requirements and might ignore fundamental aspects. Therefore, in this work we focus on software as a research outcome.

[15] http://www.icse-conferences.org/

3. Challenges

Before exploring the application of the FAIR principles to research software in detail, we want to recap current challenges that researchers face when trying to find or re-use software. For this we use results from the FAIRsFAIR survey presented in D2.1 - the Report on FAIR requirements for persistence and interoperability 2019 (Lehväslaiho et al., 2020). It was conducted in the summer and fall of 2019. It included the following free text question:

> What **challenges** do researchers in your community encounter when trying to:
> A. **find** relevant research software on the web
> B. **re-use** relevant research software on the web

There were 66 answers to the survey and all the data is available on Zenodo: 10.5281/zenodo.3518922. The participants' identified themselves with the following roles: research support staff(28); repository staff(19); research infrastructure operator (17); researcher (22), policy make(5); and other(5). Note that this question was a multi-value question and that the option research software engineer wasn't proposed and wasn't given as an "other" response.

We classified the answers to the survey into six different challenges categories, which are detailed separately:

Software dependencies and environment - technical challenge
Documentation
Accessibility & Licensing
Time and skill
Quality control
Software sustainability & management plan

3.1. Software dependencies and environment - technical challenge

The difficulty is to identify the current or desired version and which other elements (dependencies, interpreters, libraries, etc.) are needed to understand and execute the

software. The links to these elements, if they exist, can be easily broken and maintaining the stack trace is a challenge.
A piece of software can be hardware specific and finding or emulating the hardware can be a vast challenge. With that in mind, the operating system can be obsolete or difficult to access (freely), which can prevent from reusing software or reproducing a software experiment. Increasing the portability to other platforms with emulation of environments and hardware can be a solution.

3.2 Documentation

Finding the documentation and manuals, can be a challenge when encountering software. The level of detail on how to use, install, compile, execute or host the software depends on the developer's skill (and time). Multiple comments in the survey reflect the notion that lack of documentation or low levels of support make it a challenge to re-use relevant software.
There is a vital need to have a more detailed, well written documentation with examples of workflows and usage.

3.3 Accessibility & Licensing

To access and reuse software, licensing is crucial. However many software records lack in clarity on which license applies to a piece of software. Software is sometimes unavailable when it is under commercial license and the source code isn't available for inspection or when it is provided as supplementary material of a publication which can be behind a paywall. It is also unclear how open the license is. Moreover, there should be training to developers who release software and to researchers that want to use software on how to consider licensing and open source licenses.

3.4 Time and skill

Even if the software is findable, the challenge is to understand it properly so the user can install it, use it, debug it or integrate to other existing workflows, which requires time and skill. With limited time the barrier to re-use is high, and it's easy to be less motivated to redo it yourself. Also, training is needed to help researchers or research support staff produce software resources that are citable and FAIR.

3.5 Quality control

In the case of software quality, knowing which software is adequate, tested and maintained is usually information that transfers with *word of mouth. However,* evaluation of software for replication and reproducibility is necessary to guarantee quality of the research resources.

3.6 Software sustainability & management plan

Lack of a sustainability or management plan can reduce the way users can understand and reuse software. This may be seen as lack of proper/uniform description and metadata which should be uniformed and standardized in a management plan. It is sometimes unclear what is the level of support and if the software is maintained. Contact information and support information are hard to find.

3.7 Metadata

Many responses have included the lack of metadata, metadata standards or the use of loose standards as a challenge when searching and reusing software. Metadata completeness is hard to get when researchers do not use a standard and there is no consensus about what information is necessary.
Also many mentioned the lack of citation which in turn reduces the chance of discovery and reuse.

4. FAIR analysis of research software guidelines

In this section we review and analyze literature addressing the challenge when it comes to applying the FAIR principles to software or related subjects, including software citation, software curation and the place of software in academia. We collected a panorama of perspectives, from researchers, organizations and community efforts, providing a good comparison between the FAIR principles and existing software guidelines.

We propose a meta-analysis of each FAIR principle with a mapping including the exact citation from all the resources that mention the principle and how to apply it to software.

We followed a methodology described below, where we evaluated the number of times the resources referred to a FAIR principle or a similar principle that aimed for the same goals. To the best of our knowledge, there are no established assessment frameworks to measure software FAIRness, therefore we focus on a selection of related articles and guidelines. The resources used for the mapping are:

1. **Towards FAIR principles for research software (Lamprecht et al., 2019)**
   This article was published in the Data Science journal, issue 'FAIR Data, Systems and Analysis' aiming on translating the FAIR principles to research software. Their effort is supported with two case studies, along with recommendations for rewriting the FAIR principles to make them more applicable to software.
2. **"5 recommendations for FAIR software" from the Netherlands eScience Center and DANS**[16] These are straightforward guidelines for researchers on how to make software FAIR, which are available on a dedicated website to help researchers with their own software.
3. **Software citation principles (Smith et al., 2016)**
   This article published in PeerJ Computer Science is a result of the FORCE11 Software Citation Working Group, defining high level principles on software citation.
4. **RDA Software Source Code Interest Group (SSC IG) P13 activity translating FAIR principles to software**[17]
   This resource is an ad-hoc activity conducted during the RDA P13 SSC IG session, where participants were asked to map the existing FAIR principles for data to possible principles for software. Participants were asked to add items that are not in the FAIR principles.
5. **From FAIR research data toward FAIR and open research software (Hasselbring, 2020)**
   This article was published in the journal IT - Information Technology and aims at translating the FAIR principles to research software and producing a list of recommendations based on the FAIR principles and other resources.
6. **Attributing and Referencing (Research) Software: Best Practices and Outlook From**

[16] https://fair-software.nl/
[17] https://www.rd-alliance.org/rda-p13-activity-summary-applying-fair-software-dated-avril-2019

Inria (Alliez et al., 2019) This article was published in IEEE Computing in Science & Engineering aiming to analyze the existing practices handling research software at the Inria research center and providing recommendations to the academic community.

7. **Software vs. data in the context of citation (Katz et al., 2016)**
   This article is a PeerJ preprint, which details the differences between software and data, and providing simple recommendations for software citation.
8. **The science code manifesto[18] (Barnes et al., 2011)**
   This is an online manifesto, published in 2011 by the Climate Code Foundation. It was endorsed by 1227[19] researchers and organizations. It proposes five principles to reform scientific software in institutions.
9. **CoSO Opportunity Note: Encouraging a wider usage of software derived from research by The Committee for Open Science's Free Software and Open Source Project Group (Clément-Fontaine, 2019)**
   This is a committee note from the French National Open Science committee declaring the importance of software in Open Science and formulating recommendations to encourage and promote better practices for handling software in institutions.

4.1 Methodology

For the analysis, the *exact text used in the original resource* is included, to preserve the original semantics and pragmatics when reviewing the differences and similarities between translations.

At the end of each mapping, we evaluate the principle on a set of criteria, and for each we assess whether the principle appeared in the literature and if the literature supports the criteria on the particular principle. If widely referenced, this is a match. We will not seek to analyze all atomic requirements on each principle (e.g separate unique and persistent identifiers).

The key criteria identified for the analysis and evaluation of each principle are the following:

- **Relevant** - is this principle seen to be relevant to software by being frequently mentioned in the proposed resources? [without taking into account the independent dimensions]
- **Achievable** - is this principle seen to be achievable when it comes to software?
- **Measurable** - is this principle seen to be measurable on software artifacts?
- **Benefits** - is this principle seen to be useful and benefits the software resource?
  - Quality curation of the software resource
  - Recognition of software in scholarly communications

FAIR	Relevant	Achievable	Measurable	Benefits
FAIR principle				

[18] http://sciencecodemanifesto.org/
[19] Extracted on 16.9.2020 from http://sciencecodemanifesto.org/endorse

Each criterium is evaluated on the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

With this analysis, an assessment of the applicability of a principle to software artifacts can be proposed on the path to defining the FAIR principles for research software, which in itself is out of scope for this report. The scale provides a way to identify if the principle was seen in the literature and if the principle's concept was seen in a small subset, a medium subset or a large subset. We have not found disagreeing comments on the principles, however the interpretation of a few principles seems very different from the intentions of the original FAIR guiding principles.

Note that the criterium "measurable" was more difficult to find in the resources and in some cases the measurability of the principle was not commented. In some cases the resources provided a means to achieve the requirement with slight distinctions in preference that can be taken into account as a measurable principle.
The complete analysis is available in Annex B, while the summary of the analysis is presented in Section 4.2.

4.2 Compendium of FAIR software analysis

	FAIR	Relevant	Achievable	Measurable	Benefits
1	F1. (meta)data are assigned a globally unique and eternally persistent identifier.	***	***	**	***
2	F2. data are described with rich metadata.	***	**	N/A	***
3	F3. metadata specify the data identifier.	***	**	*	**
4	F4. (meta)data are registered or indexed in a searchable resource.	***	**	*	**
5	A1 (meta)data are retrievable by their identifier using a standardized communications protocol.	***	***	N/A	***
6	A1.1 the protocol is open, free, and universally implementable.	**	**	N/A	**
7	A1.2 the protocol allows for an authentication and authorization procedure, where necessary.	N/A	N/A	N/A	N/A
8	A2 metadata are accessible, even when the data are no longer available.	**	N/A	N/A	*
9	I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.	***	**	**	***
10	I2. (meta)data use vocabularies that follow FAIR principles.	**	**	**	**
11	I3. (meta)data include qualified references to other (meta)data.	**	*	N/A	***
12	R1. meta(data) have a plurality of accurate and relevant attributes.	***	***	**	***
13	R1.1. (meta)data are released with a clear and accessible data usage license.	***	***	***	***
14	R1.2. (meta)data are associated with their provenance.	***	**	*	**
15	R1.3. (meta)data meet domain-relevant community standards	***	**	*	**

4.3 Other insights and recommendations from the literature

In the following table we have gathered the recommendations and items that can't be readily mapped to an existing FAIR principle. These items focus on the following aspects:

- Interoperability by including dependencies and execution environment
- Usage of version control systems to track changes
- Credit and attribution
- Testing

Resource	Content
Towards FAIR principles for research software	*I4S- Software **dependencies are documented** and mechanisms to access them exist. [Newly proposed]* "The present tendency to package software and its dependencies, either in virtual environments and/or software containers, alleviates the practical concerns for the final user, and simply moves the issue to the generation of those packages." "...software dependencies need to be clearly documented in a formal, accessible, machine-readable, and shared way, and formally described following each programming language format."
"5 recommendations for FAIR software"	*Another suggestion to add: **Version Control** (perhaps as a specification for R1.2?)*
Software citation principles	***Credit and Attribution**: Software citations should facilitate giving scholarly credit and normative, legal attribution to all contributors to the software, recognizing that a single style or mechanism of attribution may not be applicable to all software.*
SSC IG P13 activity	*detailed documentation* *S- for Sustainability,* *P for Preservation,* *T- for Trust*
From FAIR research data toward FAIR and open research software	*"If we would achieve that software citations appear somehow very close to publication citations, this could help with giving appropriate credit and recognition for researchers who develop and maintain research software."* Note: authors highlight that in some disciplines, source code itself is data and thus the FAIR principles can apply; computer science could address some of the challenges as a research topic in their field; the authors map the FAIR principles only on a very high level but focus on hands on recommendations to make software FAIR. *"Adequate **documentation** is important, but so are **engineering practices** such as providing **testing frameworks** and **test data** for **continuous integration** to ensure that future adaptations can be tested to ensure that they work correctly."* *"Use software virtualization techniques such as Docker to support reusability across platforms"*
Attributing and Referencing (Research) Software	*Recommendation #2: Putting human at the* *heart of the evaluation. We strongly suggest to refrain, for research software, from trying to generate software **citation and credit metadata**, and in particular the list of (main) authors, using automated tools: we need instead **quality information** in the scholarly world,*
Software vs. data in the context of citation	N/A
The science code manifesto	***Credit**: "Software contributions must be included in systems of scientific assessment, credit, and recognition."* ***Citation**: "Researchers who use or adapt science source code in their research must credit the code's creators in* *resulting publications."*

	***Curation**: "Curators must provide a means of reporting and recording software defects and issues, and for communicating those defects to authors and readers. "*
CoSO opportunity note	*"A well-adapted citation mechanism needs to be constructed to make sure the visibility and reputation of researchers take the time they spend producing software into account."* Recommendation n° 4: Construct a consensual definition of a "contribution" to research software. Recommendation n° 5: Build tools which integrate this notion of a contribution to be able to effectively credit authors/designers for their software contributions.

4.4 Summary of Findings

From the literature review, we can conclude that there is a gap between the FAIR principles for data and software guidelines that can have a FAIR impact on software. Consequently the ability to assess the FAIRness of software will be different from data in parts.. The gap is mostly about the interpretation of the principles, which might benefit from different wording when it comes to software. As seen in the litterature *reuse* and *interoperability* has a different meaning when it comes to software, which is not captured in the *FAIR guiding principles*.

We observe that 10 principles are seen to be relevant by many resources and from that we conclude that these principles are highly relevant for software. However not all are beneficial for improving the software as a resource. Most principles are achievable, when it comes to software. Only one principle has a clear consensus on all the criteria, namely R1.1: releasing software with a clear and accessible license is seen to be relevant, achievable, measurable and beneficial for software.

A few recommendations, which are not included in the FAIR guiding principles (Wilkinson et al. 2016), are repeated for software, as presented in subsection 2.4. We listed the four major themes:

- Credit and citation
- Curation and quality information
- Dependencies and documentation that can be part of interoperability
- Reproducibility

In (Alliez et al. 2019) three major challenges are identified when it comes to solving the reproducibility crisis:

1. Availability: In many cases when mentioning software there is no reference to the specific version of the software and/or a way to access its source code.
    a. The availability of the software is not the same as accessibility, even if access is implied.
    b. Only archival and preservation of the software ensure availability for the long term.

2. Dependencies: it is difficult to characterize, collect and reproduce the full stack of the software's environment
    a. The dependencies and environment challenge can be included as an interoperability item or a reusability item. This challenge is specific to software and is not mentioned in the FAIR guiding principles (Wilkinson et al, 2016). However, (Lamprecht et al., 2019) state "accessibility, interoperability and (re)usability are intrinsically connected for research software", and those include aspects of installation instructions, software dependencies, and licensing as part of the extended principles. Clearly, the meaning of *interoperable* or *reusable* in a software context is different from the meaning described in the *FAIR guiding principles.*
3. Solving all at once: aggregating and archiving objects of different nature, while solutions for each class of objects exist, risks the understanding of each object's special characteristics
    a. Which means that we should carefully address each class of objects separately also when writing recommendations or principles, but more specifically when using infrastructures that archives and aggregates research outputs.

In (Hinsen, 2019), the reproducibility challenge is described in the form of a **software collapse** where software is built in layers and the upper layer depends on all the layers below. When one of the layers ceases to be maintained, it puts the software above it in jeopardy. It is important to remember that software isn't created in isolation and it is impossible to ask researchers to insure reusability for the long term when they have no control on the software stack on which their software is built.

5. Towards FAIRness of software

In this section we provide a brief panorama of mechanisms, components and infrastructures that can improve the FAIRness of software in the scholarly ecosystem. We will also describe existing software training for research on how to improve FAIRness of research software.

5.1 Existing mechanisms and components for software

5.1.1 Software identification

The first step toward FAIR research software is identification, yet software objects are very complex with different levels of granularity and different instances of the same concept, which makes it very difficult to correctly or completely identify.

Software evolves rapidly and is usually constructed on top of other software layers (environment, dependencies, etc.) The software project is not a digital object, it can be decomposed into software modules and software versions, which have a digital manifestation in software source form or as executables. In (RDA/FORCE11 Software Source Code Identification WG, 2020), an analysis of identification targets and relevant identification schemes is presented. The main conclusion from the joint SCID WG output is that software should be identified with both extrinsic identifiers for the metadata and intrinsic identifiers for the artifact, which can be at different levels of granularity (from a snapshot of a repository, a release, a directory, a file, to a code fragment).

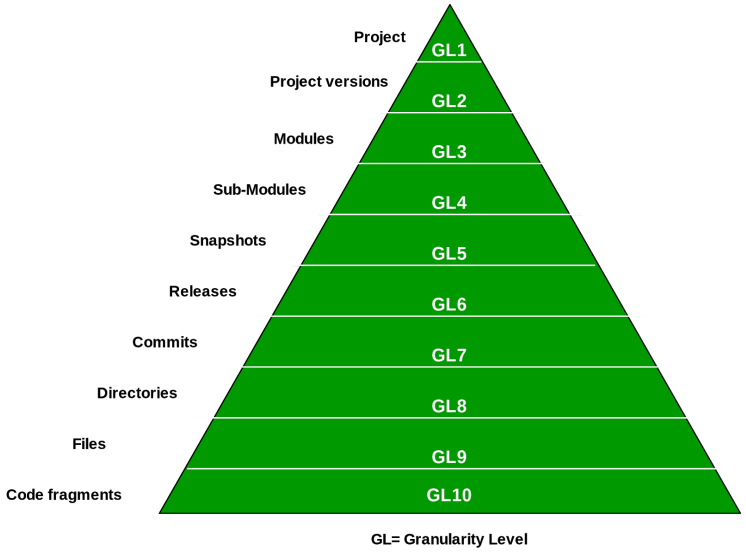

Figure 3: The granularity level diagram from the SCID WG output (RDA/FORCE11 Software Source Code Identification WG, 2020)

Intrinsic identifiers are essential when it comes to software identification in industry. They are the basis of the current practices of software development, especially with version control systems (e.g git, svn, mercurial, etc.) (Di Cosmo et al., 2020b).
With this in mind, assigning persistent identifiers to software is much more tricky than it seems. One can deposit metadata and code into an archiving service (e.g Zenodo, HAL) and get an identifier for this exact instance of the version with the coupled set of metadata. It is evident that the software will evolve and change much faster than the metadata, which won't be visible on these deposits. Software Heritage provides an archiving service that captures the entire development history and provides a PID which is an intrinsic identifier that can identify each and every element of the source code with integrity (called the SWHID[20]). SWHIDS can identify all granularity levels from snapshots (GL5) to fragments of code (GL10). In the intrinsic identifier's case only the content is identified and again the metadata should be registered in a registry, linking the metadata record to the archived content.

5.1.2 Software metadata and vocabularies

The software metadata landscape is rich with many vocabularies and ontologies. A software ontology is a classification of categories describing a software artifact with explicit specifications of its entities and relationships in a certain domain of use.

To illustrate, in many software ontologies, a software has a name and has a version number, it has authors and can depend on other software. The properties are mainly the same in most ontologies, but the terms used to describe it are different and some terms are dedicated to specific domains. For example, the relatedPublications property is only used in the academic domain because of the nature of the property.
Many general schemes have created an entity for software, but might be less detailed with software specific properties (e.g., dependencies). An overview of a few metadata schemes or vocabulary that can be used for software is presented in Figure 4, where specific software schemes and general schemes are differentiated. Each vocabulary is also liked to its ecosystem:

- digital preservation;
- linked data;
- catalogs / registries;
- and the scholarly ecosystem

[20] https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html

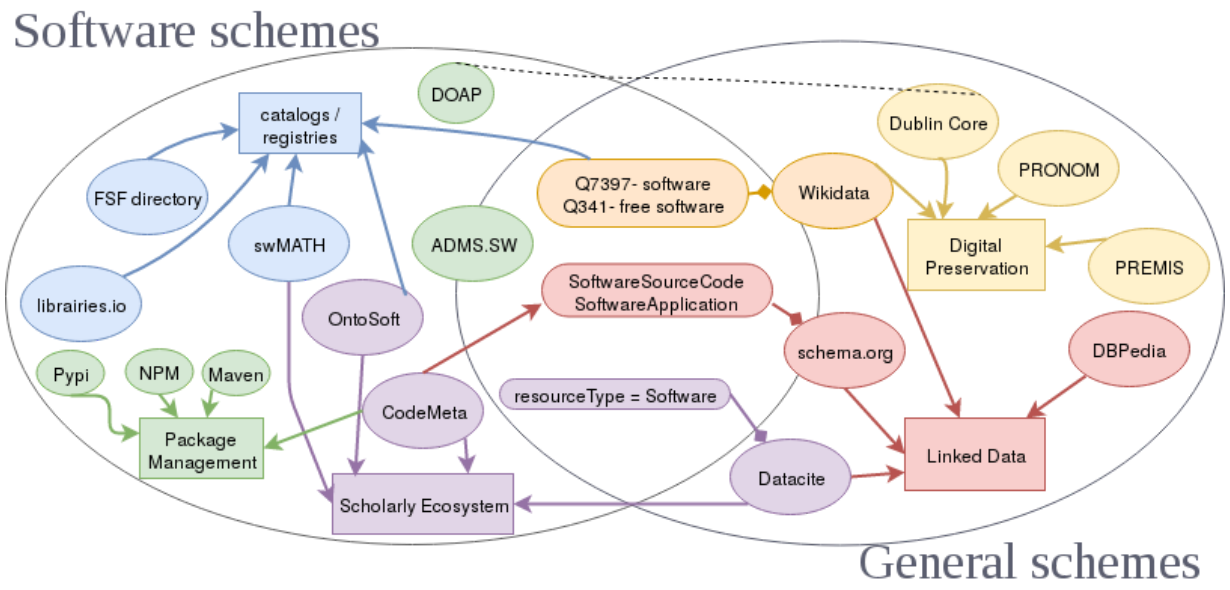

Figure 4: Software ontologies landscape from Pathways for Discovery of Free Software (slide deck from LibrePlanet 2018). (Gruenpeter & Thornton, 2018) CC-by-4[21]

The CodeMeta initiative created a concept vocabulary that can be used to standardize the exchange of software metadata across repositories and organizations (Jones et al., 2016). It was a result of the FORCE11 Software Citation working group.

In Figure 4 the CodeMeta vocabulary is connected to different metadata categories thanks to its innovative approach aggregating linked data and research metadata in an intrinsic metadata file captured inside the content. The CodeMeta initiative provides guidance for research software developers by using a specific codemeta.json file for documenting descriptive metadata in an intrinsic metadata file, to facilitate software citation. The vocabulary is built over the schema.org classes SoftwareApplication and SoftwareSourceCode which links the data for semantic web discovery. In addition, the CodeMeta crosswalk table can facilitate translation between ontologies and metadata standards for software, the Rosetta stone of software metadata.

5.1.3 Software licenses and SPDX

Software licensing is a well established practice in industry and in the Free and Open Source Software (FOSS) community, because software is protected under copyright law (Ballhausen, 2019). FOSS licenses are an instrument to use and share software, under the four essential FOSS freedoms: run, study, distribute copies and redistribute copies of modified code.

[21] https://en.wikipedia.org/w/index.php?title=File:Pathways-discovery-free.pdf&page=31

The Software Package Data Exchange (SPDX) Specification is a standard format to communicate about software components, licenses and copyrights[22]. SPDX is the standard used by many organizations and is adopted in industry and in the FOSS community when agreeing on naming licenses.

The Reuse project[23], which was started by the Free Software Foundation Europe (FSFE)[24] is complementary to SPDX, it facilitates the application of a license to software, with three steps:

1. Choose license
2. Add license
3. Confirm REUSE compliance

5.1.4 Software Curation

There are curation mechanisms in place that provide a controlled workflow where software is moderated by digital archivists and curate the metadata of the software. This aims to provide a better quality metadata and a curated software artifact. In Figure 5 an example of this workflow from HAL - the french national archive.

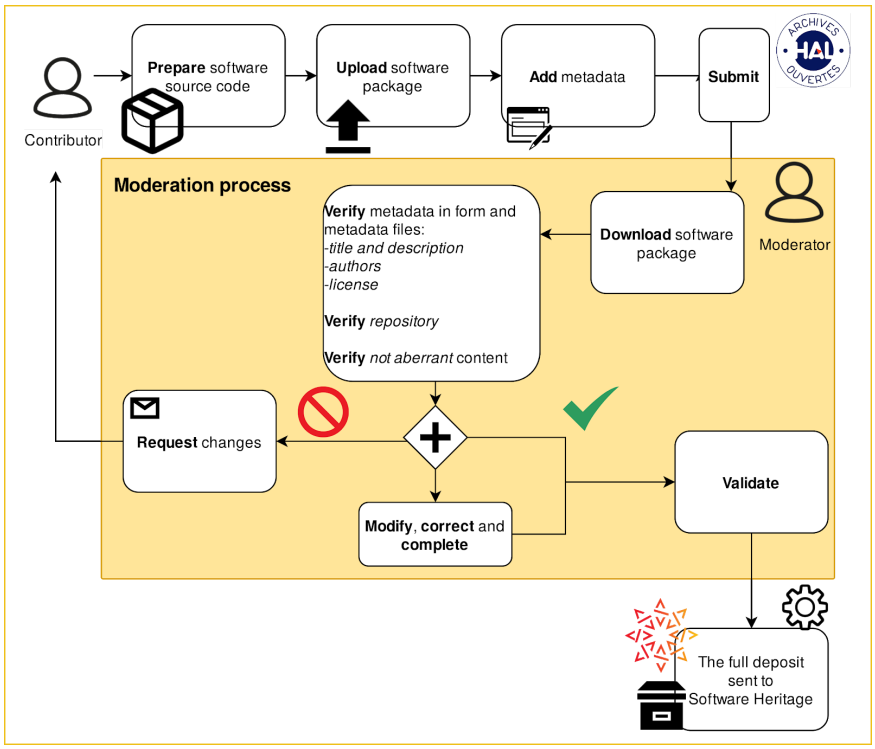

Figure 5: The moderation process of software deposits in HAL from (Di Cosmo et al., 2020a)

[22] https://spdx.dev/about/
[23] https://reuse.software/
[24] https://fsfe.org/

In (Di Cosmo et al., 2020a) and in (Alliez et al. 2019), keeping humans in the research dissemination workflow is seen to be essential to have a quality record of the software resource.

4.1.5 Software artifact evaluation and badging

There are different initiatives that tried to improve the quality of the published artifact with artifact evaluation and badging, see the Artifact Evaluation Committee (AEC)[25]. One example is the ACM artifact evaluation, giving an incentive to follow FAIR principles with research software. It defined a set of qualifiers for research artifacts, in order to verify fundamental requirements to achieve reproducible research. The NISO Taxonomy, Definitions, and Recognition Badging Scheme Working Group has considered a number of these schemes and proposed a Recommended Practice on Reproducibility Badging and Definitions[26] which includes aspects relevant to the FAIRness of software, including the availability of software outputs and related metadata.
Artifact evaluation is one step towards improving software quality and solving the reproducibility challenge.

Figure 6: The ACM badges for artifact evaluation[27]

5.2 The landscape of existing infrastructures

There are different infrastructures already in place that provide services and guidance for researchers, which are actively developing and creating software for their research. Many

[25] https://www.artifact-eval.org/
[26] Draft Recommended Practice at: https://groups.niso.org/apps/group_public/download.php/23561/RP-31-202X_Reproducibility_Badging_draft_for_public_comment.pdf
[27] https://www.acm.org/publications/policies/artifact-review-and-badging-current

researchers are not aware of the possibilities they have to improve the findability and reusability of their software, which at the same time can offer them credit for their work.

In this section we briefly present key infrastructures for research software in three major types of infrastructures:

- Archives and institutional repositories
- Journals and publishers
- Registries / indexers / aggregators

We are at a turning point where one one side the community understands how much software is important and on the other we have infrastructures that can support software. It is time that policy makers, institutions, universities, international initiatives and other key actors that are involved in making recommendations for software will be aware of the existing infrastructures and their contribution to the scholarly ecosystem.

5.2.1 Software archives and institutional repositories

Software archives and Institutional Repositories (IR) are fundamental infrastructures providing access to software on the **long-term**. It is important to note the differences between the existing archives and IR. Many are transversal repositories accepting different types of digital objects, but in turn they provide PIDs. In very few cases, we can find specific software metadata.

Alongside the traditional IR, a new archive has emerged this past few years, the universal source code archive - Software Heritage (SWH). It aims to collect, preserve and share all software source code publicly available (Di Cosmo & Zacchiroli, 2017). With SWH, it is possible to use the `save code now` functionality to preserve a complete snapshot of a git (or other version control system) repository the same way web pages can be saved with the wayback machine[28].

Clearly, preserving the software resources for research is a core element to answer many aspects of the FAIR principles for software. Software archiving does not necessarily mean the software must be published, but it should be publicly available in a permanent location.

In annex C.1, we present two case studies for archives: Zenodo and Software heritage.

5.2.2 Software journals and publishers

One of the important challenges for researchers that create software is getting credit for their work. A few journals have provided different solutions for these researchers. JOSS[29], eLife[30],

[28] archive.org
[29] https://joss.theoj.org/
[30] https://elifesciences.org/

IPOL[31], JORS[32], SoftwareX[33] and others allow or require that the software source code will be peer reviewed and published with the article.

By introducing peer review of software source code, there is a level of certification and recognition of the reviewed artifacts.

In annex C.2, we present SoftwareX as a case study for software journals.

5.2.3 Software registries / indexers / aggregators

Findability of software is only possible if it exists in a public location and is fully described with a complete and semantically acceptable metadata vocabulary. Registries, indexers and aggregators provide search mechanisms to find software projects.

A few precious examples exists that are discipline specific, like ASCL[34] for astrophysics software (Allen and Scmidt, 2015) and swMath[35] for mathematical software (Bönisch et al. 2013, Chrapary et al. 2017).
Note that there are also package managers that can play the role of a registry without the academic focus, for example CRAN[36] for software packages in R or PyPI[37] for software packages in Python.

In annex C.3, we present swMath as a case study for software registries.

5.2.4 Research software training

Research software training is the glue between the building blocks of the FAIR ecosystem. Without training we can continue recommending and reporting without it taking any affect and the research lifecycle or on the researcher recurrent activities.

We should acknowledge the organizations, institutions and universities that invest in research software training and provide the researcher with the basic knowledge of the FAIR principles and the existing infrastructures and services that can help improve their research.

The Carpentries[38] is an example for a software training organization aiming to provide information and tools for researchers to do more efficient, open and reproducible research.

In annex C.4, we present The Carpentries as a case study for research software training.

[31] https://www.ipol.im/
[32] https://openresearchsoftware.metajnl.com/
[33] https://www.sciencedirect.com/journal/softwarex
[34] https://ascl.net/
[35] https://swmath.org/
[36] https://cran.r-project.org/
[37] https://pypi.org/
[38] https://carpentries.org/

6. Recommendations

To conclude this report we present a set of recommendations for the creation of FAIR guiding principles for research software. These recommendations are based on the exploration of the role of software in the FAIR ecosystem, the analysis of the literature on applying the FAIR principles to software, and the review of existing solutions and infrastructures presented in the sections above.

The following recommendations are high-level requirements in the next practical steps toward a community effort defining the FAIR principles for research software. We specifically have in mind the joint RDA, FORCE11 & ReSA FAIR4RS WG, which is working on defining the FAIR principles for research software and intends to involve different communities for the adoption of the new principles. The recommendations in this report may serve as a useful starting point to scope and organize the FAIR4RS WG work. Each recommendations has a requirement level, as defined in RFC2119[39]:

- MUST is an absolute requirement
- SHOULD is a needed requirement for which exceptions are possible
- MAY is an optional requirement

At a more general level, it is to be acknowledged that any new principle may lead to extra requirements enforced on researchers, who are already facing significant challenges when developing or maintaining software, which is a complex and living object. The time and effort required to abide by these principles must hence be properly taken into account; to find a proper balance between effort and return we suggest that a large community be consulted. In order to maximize adoption, clear and immediate benefits should be offered to the researcher, e.g. by reducing the amount of times she is requested to enter the same information in different systems in different phases of her career.

[39] https://tools.ietf.org/html/rfc2119

Recommendation n°1	FAIR principles for research software outcomes **MUST** be produced by taking into account the specific nature of software and not as just a simple adaptation of the FAIR guiding principles for data.
Recommendation n°2	Applying principles and recommendations to software demands effort, time and skill. The realistic nature of these principles **MUST** be considered.
Recommendation n°3	A large community forum **MUST** be consulted when writing the principles. This community forum **MUST** include stakeholders from different disciplines and with different roles, looking at software in all its aspects: as a tool, as a research outcome and as the object of research.
Recommendation n°4	Existing infrastructures that already provide solutions for software artifacts **SHOULD** be asked to review the FAIR principles for research software.
Recommendation n°5	Each principle **MUST** be relevant for software source code.
Recommendation n°6	Each principle **MUST** be achievable for software source code.
Recommendation n°7	Each principle **SHOULD** be measurable for software source code; detailed explanations of how a measurable principle is measured MUST be available.
Recommendation n°8	Each principle **SHOULD** contribute to software recognition in scholarly communication.
Recommendation n°9	Each principle **SHOULD** contribute to the curation quality of the software resource.
Recommendation n°10	Each principle **MAY** solve one or more research software challenges (e.g credit, reproducibility, sustainability & management, documentation, quality control, quality metadata, licensing and more).

7. Conclusion

This report is part of the FAIRsFAIR project's outputs and it is the first milestone focused specifically on software as a digital object. Software has an important place in academia and as such it has an important place in a FAIR ecosystem (European Commision, 2018). However, and notwithstanding its importance, there is no widely accepted definition of the FAIR guiding principles to research software (as opposed to research data). This report aims to help close this gap by bringing together a review of the role of software in the FAIR ecosystem, a survey of current issues and pain points, an in-depth analysis of existing resources for FAIRification of software, a discussion of existing solutions and infrastructure, and - finally - a set of recommendations for the creation of community-supported FAIR guiding principles for research software.

We have discussed the distinct roles of software which is produced and consumed throughout the research lifecycle, which can be concisely resumed with the following functionalities:

- as a tool;
- as a research outcome or result; and
- as the object of research.

We have also discussed the complexity of software and its roles in a FAIR ecosystem, in order to emphasize the requirement to evaluate FAIRness of software differently, depending on the purpose it fulfills. Trying to answer all three at once can impose unnecessary requirements and might ignore fundamental aspects. In addition, we illustrated the complexity of software using the granularity level definition that was introduced in the recent joint RDA & FORCE11 software identification working group output (RDA/FORCE11 Software Source Code Identification WG, 2020). Given this complexity, focusing on source code is necessary when it comes to the FAIRification of research outcomes.

We have also analyzed nine current resources that call for the recognition of software in academia and present guidelines or recommendations to improve software status, by becoming FAIR or by improving curation of software in general. Throughout this analysis we have looked for demonstrations of each FAIR principle to see if the principle is relevant, achievable, measurable and benefits software. From the analysis we captured the gap between the FAIR guiding principles for data and the aspiring guidelines needed for software as a research outcome.

Furthermore, we have listed and interpreted the recurrent challenges that researchers and research infrastructures face today when handling software, which include technical as well as social aspects; and we have presented a number of services and infrastructures that support

software in academia, and that could form the building blocks of a FAIR ecosystem that includes software, both as research outcomes as well as objects of research.

In the last section above, we propose 10 recommendations to follow when creating guidelines, or more specifically, when considering the application of the FAIR principles to research software.

As a final consideration, the authors would like to underline how, in academia as in the FAIR ecosystem, it is crucial to consider software as a first class citizen and provide guidelines, recommendations, metrics, solutions and infrastructures that acknowledge the importance of software while adequately respecting the differences between software and other digital objects.

8. Acknowledgements

We would like to thank the following people for providing valuable input through comments and discussions on version 1.0 of this report as a first experts review of the report :
Paula Andrea Martinez, Dan S. Katz, Neil Chue Hong, Michelle Barker and other members of the FAIR4RS WG steering committee.

Annex A: FAIRsFAIR Task 2.4 Statement of Work

FAIRsFAIR Task 2.4 (T2.4), as stipulated in the FAIRsFAIR project proposal, is tasked with "*extending the FAIR concept currently applied to data to the range of data services needed to enable and support FAIR data, and to software"*. This ambition naturally splits into two related but distinct topics, namely (i) 'FAIRness of services' and (ii) 'FAIR principles for (research) software'. T2.4 will be working on these topics alongside each other, seeking synergies where possible but also mindful of intrinsic differences that warrant a parallel approach.

In terms of the first of these two topics, 'FAIRness of services', T2.4 will be considering how services can make data (more) FAIR. This formulation respects and builds upon the original FAIR principles, which were articulated specifically for research data objects (and not for services or software). Taking these principles as a starting point, **T2.4 will be delivering an assessment framework that can be used to gauge how a given service acting on a data object makes that data object 'more FAIR', 'less FAIR' or 'equally FAIR'**. In formulating such a framework, it is anticipated that some of the FAIR principles may apply to services as they do to data objects (e.g., "being registered and indexed in a searchable resource" under Findability). Equally there will be FAIR principles for data objects that do not translate to services, and there will be criteria for services that do not directly map onto one of the original FAIR principles (e.g., quality measures or warranties specific to services such as availability or trustworthiness). In other words, the directive of this task goes beyond a naïve mapping of the FAIR principles for data objects to services; rather, it aims to support an optimal interplay between services and research (data) objects to realize a 'FAIR ecosystem' (as articulated in (4)).

Two remarks are in order. First, the proposed assessment framework will be normative in the sense that it scores against a desired future state, i.e. it is constructed relative to a set of desired features and qualities for services to have. The task will thus also provide concrete, actionable recommendations for services to increase their level of 'FAIRness'. Such desiderata will be defined from community input about the current state (including good practices and current pain points), desired state, and recommendations to close the gap between current and desired state. The second remark is about scope: The task considers all services that create, read, update or delete data at any point in the data life cycle[40]. For the sake of focus, it will

[40] As working definition for the concept of a 'service' in the context of research data, we will adopt the formulation put forward by the ICSU-WDS/RDA Publishing Data Workflows WG (15): "A means of delivering value to the producers and users of digital objects by facilitating outcomes they want to achieve without the ownership of specific costs or risks"

primarily concentrate on digital services with a strong IT component, i.e. strongly relying on technology to deliver value users.

The second objective of the task pertains to research software, i.e. software artefacts that are the output of a research activity. **T2.4 will deliver recommendations on how to apply or adapt the FAIR principles, formulated for data objects, to software artefacts.** Here it is expected that a naïve mapping of the original FAIR principles may already provide a useful starting point (in that such a straightforward application "may act as a guideline for those wishing to enhance the reusability of their research software holdings", wording adapted from (2)); yet adaptations and/or extensions will likely be required to account for the special nature of research software (such as its dynamic nature with large numbers of versions and library dependencies). Also for this objective, community input about the current state, desired state and recommendations to close the gap will be central in formulating the recommendations.

While the subject matter and intended approach for both objectives within T2.4 have their differences, they are joined at the hip by the overarching ambition to create a 'FAIR ecosystem' which comprises of FAIR Digital Objects - including data and software - together with relevant services and infrastructure(4). This suggests that there will be ample connections and inter-related questions between the objectives, which justifies addressing them in parallel within the task.

The explicit connection between T2.4 and the notion of a 'FAIR ecosystem' also signals that the work carried out by the task will not stand in isolation. T2.4 will seek coordination and collaboration with a number of relevant projects and organization, including (but not limited to) the EOSC FAIR WG, GO FAIR, FAIRsharing, the joint FORCE11 & RDA Software Identification WG, the RDA Software Source Code IG, and FAIRsFAIR WP4 around FAIR certification.

Finally, the approach taken within T2.4 will be guided by the ambition to deliver concrete, reasonable and actionable outputs that are rooted in real-life problems and ready to be adopted by the various stakeholders – and hence will err on the side of 'progress' over 'perfection'.

Milestones & Deliverables

M12	Feb 2020	Milestone M2.7	Assessment report on FAIRness of services
M19	Sep 2020	Milestone M2.15	Assessment report on FAIRness of research software *(present report)*

M20	Oct 2020	Milestone M2.10	Report on basic framework on FAIRness of services
M30	Aug 2021	Deliverable D2.7	Framework for assessing FAIR services

Annex B: Complete analysis of software guidelines

B.1 Findable

F1. (meta)data are assigned a globally unique and eternally persistent identifier.

Resource	Content
Towards FAIR principles for research software	**Rephrased**: *"Software and its associated metadata have a global, unique and persistent identifier for each **released version**." "Software versions should get assigned different PIDs as they represent specific developmental stages of the software. This is important as it will contribute to guaranteeing data provenance and reproducible research processes."*
"5 recommendations for FAIR software"	**Citation**: *"Regarding archiving copies of your software, look for services that store their own copy of a snapshot of your software, such that whatever persistent identifier you get (DOI, URN, ARK, etc) points to a specific version of the software, and will continue to resolve to exactly that version for the foreseeable future."*
Software citation principles	**Unique Identification**: *A software citation should include a method for identification that is machine actionable, globally unique, interoperable, and recognized by at least a community of the corresponding domain experts, and preferably by general public researchers.*
SSC IG P13 activity	F1- *an **identifier** for each piece of software, need to define which **unit** gets an identifier*
From FAIR research data toward FAIR and open research software	*(not explicitly discussed)*
Attributing and Referencing (Research) Software	*when looking for reproducibility, it is necessary to **precisely identify** not only the main software but also its **whole environment** and to make it available in **an open and perennial way**. In this context, we need verifiable build methods and **intrinsic identifiers** that do not depend on resolvers that can be abused or compromised*
Software vs. data in the context of citation	"best practices to facilitate reproducibility of computational science involve archiving of the following, in durable, plaintext formats: 1. the **software itself, in source code form** in a trusted digital repository ..."
The science code manifesto	*N/A*
CoSO opportunity note	*Recommendation n° 3: Promote archiving and **referencing** best practices for research software.*

FAIR	Relevant	Achievable	Measurable	Benefits
F1. (meta)data are assigned a globally unique and eternally persistent identifier.	***	***	**	***

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

F2. data are described with rich metadata.

Resource	Content
Towards FAIR principles for research software	**"Rephrased**: *Software is described with rich metadata."* *"In order for others to find and use that software, they need information about what it does, what it depends on and how it works."* *"Additionally, some programming languages provide a way to add metadata to software sources, i.e., packages"*
"5 recommendations for FAIR software"	**Registry::** *"What metadata does the community registry offer? This is sometimes described in the documentation of the registry, but you can also see for yourself by installing a tool like the OpenLink Structured Data Sniffer. "* **Citation:** : *"Regarding archiving copies of your software, look for services that store their own copy of a snapshot of your software, such that whatever persistent identifier you get (DOI, URN, ARK, etc) points to a specific version of the software, and will continue to resolve to exactly that version for the foreseeable future."*
Software citation principles	*"Necessary metadata should then be included in a CITATION file (Wilson, 2013) or machine-readable CITATION.jsonld file (Katz & Smith, 2015). "*
SSC IG P13 activity	F2- *need to better understand how sw is applied- what does it do, control vocabulary, metadata must have declared semantics and formal syntax taxonomy for licenses (spdx)*
From FAIR research data toward FAIR and open..	*Provide software metadata to improve software retrieval*
Attributing and Referencing (Research) Software	*"Recommendation #1: ... a rich taxonomy for software contributions, that must not be flattened out on the simple role of software developer"* *"The difficulty lies in getting quality metadata, and in particular in determining who should get credit, for what kind of contribution, and who has authority to make these decisions."*
Software vs. data in the context of citation	"best practices to facilitate reproducibility of computational science involve archiving of the following, in durable, plaintext formats: ... 2. structured or unstructured narrative documentation (e.g., the ODD protocol (Grimm et al., 2013)) specifically covering key components of the software..."
The science code manifesto	Curation: "*The curator must provide the specific version of software used in a publication, along with ownership and licensing information, accessible by a unique stable identifier such as a DOI or URI."*
CoSO opportunity note	*(not explicitly discussed)*

FAIR	Relevant	Achievable	Measurable	Benefits
F2. data are described with rich metadata.	***	***	N/A	***

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

F3. metadata specify the data identifier.

Resource	Content
Towards FAIR principles for research software	***Rephrased and extended:*** *"Metadata clearly and explicitly include identifiers for* ***all*** *the versions of the software it describes."* *"For reproducibility and reusability purposes, any person and/or system examining the metadata needs to be able to identify which version of the software is described by it"*
"5 recommendations for FAIR software"	*(not explicitly discussed)*
Software citation principles	**"Specificity**: *Software citations should facilitate identification of, and access to, the specific version of software that was used. Software identification should be as specific as necessary, such as* ***using version numbers***, ***revision numbers****, or variants such as platforms."*
SSC IG P13 activity	*"level of software (package? component? a piece of a large library?"*
From FAIR research data toward FAIR and open research software	*(not explicitly discussed)*
Attributing and Referencing (Research) Software	***"Recommendation #3****: Distinguish citation from* ***reference*** *It is essential to distinguish citations to projects or results from exact references to software and their environment, and we believe that both should be used in articles."*
Software vs. data in the context of citation	*(not explicitly discussed)*
The science code manifesto	*"The source code made available should be the* ***exact version*** *used in processing data for the published paper."*
CoSO opportunity note	*"Recommendation n° 3: Promote archiving and* ***referencing*** *best practices for research software."*

FAIR	Relevant	Achievable	Measurable	Benefits
F3. metadata specify the data identifier.	***	**	*	**

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

F4. (meta)data are registered or indexed in a searchable resource.

Resource	Content
Towards FAIR principles for research software	*Rephrased: Software and its associated metadata are included in a searchable software registry.*
"5 recommendations for FAIR software"	**Registry:** *Register your code in a community registry"* *"For others to make use of your work, they need to be able to find it first. Community registries are like the yellow pages for software -- registering your software makes it easier for others to find it, particularly through the use of search engines such as Google"* *"What metadata does the community registry offer? This is sometimes described in the documentation of the registry, but you can also see for yourself by installing a tool like the OpenLink Structured Data Sniffer. "*
Software citation principles	*(not explicitly discussed)*
SSC IG P13 activity	*"source code and metadata are registered/indexed in a searchable resource, software repositories need to think about long-term management,* *source code is much more searchable because it's always text, metadata is largely embedded in the code as comments (code is mostly text that can be searched easily)"*
From FAIR research data toward FAIR and open research software	*"Employ research software observatories which may serve as retrieval service"*
Attributing and Referencing (Research) Software	*(not explicitly discussed)*
Software vs. data in the context of citation	*(not explicitly discussed)*
The science code manifesto	*(not explicitly discussed)*
CoSO opportunity note	*Recommendation n° 7: "Encourage academic institutions to share research software metadata"*

FAIR	Relevant	Achievable	Measurable	Benefits
F4. (meta)data are registered or indexed in a searchable resource.	***	**	*	**

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

B.2 Accessible

A1 (meta)data are retrievable by their identifier using a standardized communications protocol.

Resource	Content
Towards FAIR principles for research software	*Rephrased: "Software and its associated metadata are accessible by their identifier using a standardized communications protocol."* *"Retrievability of research software and its metadata **can be achieved** by **depositing** it in an appropriate repository and/or registry."* *"It is worth to re-emphasize that research software are not single, isolated, digital objects"*
"5 recommendationsfor FAIR software"	*(not explicitly discussed)*
Software citation principles	*"Accessibility: Software citations should facilitate access to the software itself and to its associated metadata, documentation, data, and other materials necessary for both humans and machines to make informed use of the referenced software."*
SSC IG P13 activity	*"replace `meta(data)` with source code and metadata, active curated entries for software"*
From FAIR research data ...	*"Use repositories such as Zenodo to access archived software versions"*
Attributing and Referencing (Research) Software	*"Recommendation #3: Distinguish citation from reference It is essential to distinguish citations to projects or results from exact references to software and their environment, and we believe that both should be used in articles. We also strongly encourage the use of tools like GUIX and Software Heritage to build such perennial references."*
Software vs. data in the context of citation	*(not explicitly discussed)*
The science code manifesto	***"Code**: All source code written specifically to process data for a published paper must be available to the reviewers and readers of the paper."*
CoSO opportunity note	*"Guaranteeing permanent access to both the software and the data it manipulates means it must be possible:* ● *to refer to particular versions of the software used as well as their execution environments on a long-term basis;* ● *to possess platforms capable of permanently storing these versions;* ● *to possess hardware and system environments which allow software to be re-used identically. This is a complex scientific problem because the rapid obsolescence of hardware can have a strong impact on the reproducibility of certain types of results."*

FAIR	Relevant	Achievable	Measurable	Benefits
A1 (meta)data are retrievable by their identifier using a standardized communications protocol.	***	***	N/A	***

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

A1.1 the protocol is open, free, and universally implementable.

Resource	Content
Towards FAIR principles for research software	*"Usually software (and its metadata) can be downloaded directly from the repository and/or website via standard protocols (HTTP/SSH).* *There is no need to rephrase this specific item as it generally applies to any digital resource exposed via the web, and thus to both data and software."*
"5 recommendations for FAIR software"	***Repository:*** *"Developing scientific software in publicly accessible repositories enables early involvement of users, helps build collaborations, contributes to the reproducibility of results generated by the software, facilitates software reusability, and contributes to improving software quality. "*
Software citation principles	*(not explicitly discussed)*
SSC IG P13 activity	*(not explicitly discussed)*
From FAIR research data toward FAIR and open research software	*"Use software development platforms such as GitHub for code cloning"*
Attributing and Referencing (Research) Software	*"With the rise of Free/Open Source Software, which requires and fosters source code accessibility, access has been provided to an enormous amount of software source code that can be massively reused."* *"With the emerging awareness of the importance of making research openly accessible and reproducible, Inria has stepped up its engagement for software"*
Software vs. data in the context of citation	*(not explicitly discussed)*
The science code manifesto	Curation: "Source code must remain available, linked to related materials, for the useful lifetime of the publication."
CoSO opportunity note	*(not explicitly discussed)*

FAIR	Relevant	Achievable	Measurable	Benefits
A1.1 the protocol is open, free, and universally implementable.	**	**	N/A	**

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

A1.2 the protocol allows for an authentication and authorization procedure, where necessary.

Resource	Content
Towards FAIR principles for research software	*"The protocol allows for an authentication and authorization procedure, where necessary..[Remain the same]"* *"Similarly, it might be possible that users might need to register, and/or authenticate, before downloading binaries or, in the case of web applications, using the software. In all cases, access conditions should be justified and documented."*
"5 recommendations for FAIR software"	*(not explicitly discussed)*
Software citation principles	*(not explicitly discussed)*
SSC IG P13 activity	*(not explicitly discussed)*
From FAIR research data toward FAIR and open research software	*(not explicitly discussed)*
Attributing and Referencing (Research) Software	*(not explicitly discussed)*
Software vs. data in the context of citation	*(not explicitly discussed)*
The science code manifesto	*(not explicitly discussed)*
CoSO opportunity note	*(not explicitly discussed)*

FAIR	Relevant	Achievable	Measurable	Benefits
A1.2 the protocol allows for an authentication and authorization procedure, where necessary.	N/A	N/A	N/A	N/A

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

A2. metadata are accessible, even when the data are no longer available.

Resource	Content
Towards FAIR principles for research software	*Rephrased: "Software metadata are accessible, even when the software is no longer available."* *"Metadata provides the context for understanding research software, and this should persist even when the software itself is no longer available."*
"5 recommendations for FAIR software"	*(not explicitly discussed)*
Software citation principles	*"we recognize that the software version may no longer be available, but it still should be cited along with information about how it was accessed."*
SSC IG P13 activity	*(not explicitly discussed)*
From FAIR research data toward FAIR and open research software	*"Use research software observatories as dedicated repository services"* (not explicitly)
Attributing and Referencing (Research) Software	*(not explicitly discussed)*
Software vs. data in the context of citation	*(not explicitly discussed)*
The science code manifesto	*(not explicitly discussed)*
CoSO opportunity note	*(not explicitly discussed)*

FAIR	Relevant	Achievable	Measurable	Benefits
A2 metadata are accessible, even when the data are no longer available.	**	N/A	N/A	*

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

B.3 Interoperable

I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.

Resource	Content
Towards FAIR principles for research software	*Rephrased and extended: "Software and its associated metadata use a formal, accessible, shared and broadly applicable language to facilitate machine readability and data exchange."* *"Interoperability for research software can be understood in two dimensions: as part of workflows (horizontal dimension) and as stack of digital objects that need to work together at compilation and execution times (vertical dimension)."* *"When considering research software as part of a workflow, software should be able to share input and/or output data sets with other software."*
"5 recommendations for FAIR software"	***Registry:*** *: "What metadata does the community registry offer? This is sometimes described in the documentation of the registry, but you can also see for yourself by installing a tool like the OpenLink Structured Data Sniffer. "* ***Software quality:*** *: "Checklists help you write good quality software. What exactly constitutes 'good quality' depends on the specific application of the software, but typically covers things like documenting the source code, using continuous testing, and following standardized code patterns."*
Software citation principles	*(not explicitly discussed)*
SSC IG P13 activity	*prefer open source software when doing published research* ***formal syntax taxonomy for licenses*** *(spdx)*
From FAIR research data...	*Provide* ***proper interface definitions*** *in modular software architectures*
Attributing and Referencing (Research) Software	• *"the need of a* ***rich metadata schema*** *to describe software projects;* • *the need of a rich* ***taxonomy for software contributions****, that must not be flattened out on the simple role of software developer;* • *Last but not least, while tools may help, a* ***careful human process*** *involving the research teams is crucial to produce the qualied information and metadata that is needed for proper credit and attribution in the scholarly world."*
Software vs. data in the context of citation	*(not explicitly discussed)*
The science code manifesto	*(not explicitly discussed)*
CoSO opportunity note	*Recommendation n° 6: Promote a* ***shareable standardized metadata schema*** *for the software with a view to opening up software metadata derived from research.*

FAIR	Relevant	Achievable	Measurable	Benefits
I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.	***	**	**	***

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

I2. (meta)data use vocabularies that follow FAIR principles.

Resource	Content
Towards FAIR principles for research software	*Reinterpreted, extended and split: "I2S.1 - Software and its associated metadata are formally described using controlled vocabularies that follow the FAIR principles. I2S.2- Software use and produce data in types and formats that are formally described using controlled vocabularies that follow the FAIR principles."*
"5 recommendations for FAIR software"	***Citation:** "The CodeMeta[41] standard and the Citation File Format[42] were specifically designed to enable citation of software and will likely meet your needs. For either one, you write a plain text file with citation metadata, which you then distribute with your software."*
Software citation principles	*Existing efforts around metadata standards. Producing detailed specifications and recommendations for possible metadata standards to support software citation was not within the scope of this working group. However some discussion on the topic did occur and there was significant interest in the wider community to produce standards for describing research software metadata.*
SSC IG P13 activity	*"**commonly used** file formats"*
From FAIR research data toward FAIR and open research software	*"Conform to established **software standards**"*
Attributing and Referencing (Research) Software	*"As an illustration of this recommendation, the rich metadata collected by HAL in the deposit process are sent to SWH using the now **standard CodeMeta schema.**"* CodeMeta is an example of a vocabulary that follows FAIR principles
Software vs. data in the context of citation	*(not explicitly discussed)*
The science code manifesto	*(not explicitly discussed)*
CoSO opportunity note	*"Recommendation n° 6: Promote a **shareable standardized metadata schema** for the software with a view to opening up software metadata derived from research."*

FAIR	Relevant	Achievable	Measurable	Benefits
I2. (meta)data use vocabularies that follow FAIR principles.	**	**	**	***

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

[41] https://codemeta.github.io/
[42] https://citation-file-format.github.io/

I3. (meta)data include qualified references to other (meta)data.

Resource	Content
Towards FAIR principles for research software	*"Discarded"* *"I3 aims to interconnect data sets by semantically meaningful relationships..... However, such relationships are difficult to translate to the case of research software. We found the closest resemblance of this principle to be in software dependencies." => I4S*
"5 recommendations for FAIR software"	*(not explicitly discussed)*
Software citation principles	*(not explicitly discussed)*
SSC IG P13 activity	*(not explicitly discussed)*
From FAIR research data toward FAIR and open research software	*Use software virtualization techniques for portability* *Participate in artifact evaluation processes to evaluate interoperability*
Attributing and Referencing (Research) Software	*"First, the frequent lack of availability of the software source code, and/or of **precise references** to the right version of it, is a major issue [7]. Solving this issue (Reproducibility) requires stable and perennial source code archives and specialized identifiers [9]."*
Software vs. data in the context of citation	*(not explicitly discussed)"First, the frequent lack of availability of the software source code, and/or of precise references to the right version of it, is a major issue [7]. Solving this issue (Reproducibility) requires stable and perennial source code archives and specialized identifiers [9]."*
The science code manifesto	*"The software should be linked to a list of publications using the code, to other versions of the code, to relevant versions of tools and libraries used, and to derived code."*
CoSO opportunity note	*"it is therefore necessary to define reference methodologies for technology transfer based on existing mechanisms (....), and to share them with the actors concerned (...)."*

FAIR	Relevant	Achievable	Measurable	Benefits
I3. (meta)data include qualified references to other (meta)data.	**	*	N/A	***

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

B.4 Reusable

R1. meta(data) have a plurality of accurate and relevant attributes.

Resource	Content
Towards FAIR principles for research software	*Rephrased: "Software and its associated metadata are richly described with a plurality of accurate and relevant attributes."* (Note that this principles isn't developed)
"5 recommendations for FAIR software"	***Registry:*** *"With metadata, search engines are able to get some idea of what the software is about, what problem it addresses, and what domain it is suited for. In turn, this helps improve the ranking of the software in the search results -- better metadata means better ranking."*
Software citation principles	In Table 2. A collection of use cases and basic metadata requirements for software citation (differentiating required metadata and beneficial metadata)
SSC IG P13 activity	*needs metadata that isn't available (authorship, dependencies)*
From FAIR research data toward FAIR and open research software	*"For reusability, metadata, data and software should be well-described such that they can be reused, combined and extended in different settings."*
Attributing and Referencing (Research) Software	*"the need of a rich metadata schema to describe software projects;"*
Software vs. data in the context of citation	*"best practices to facilitate reproducibility of computational science involve archiving of the following, in durable, plaintext formats:* *...* *2. structured or unstructured narrative documentation (e.g., the ODD protocol (Grimm et al., 2013)) specifically covering key components of the software"*
The science code manifesto	*(not explicitly discussed)*
CoSO opportunity note	*(not explicitly discussed)*

FAIR	Relevant	Achievable	Measurable	Benefits
R1. meta(data) have a plurality of accurate and relevant attributes.	***	***	**	***

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

R1.1. (meta)data are released with a clear and accessible data usage license.

Resource	Content
Towards FAIR principles for research software	*Software and its associated metadata have independent, clear and accessible usage licenses **compatible with the software dependencies**. [Rephrased and extended]*
"5 recommendations"	***License:*** *"Any creative work (including software) is automatically protected by copyright. Even when the software is available via code sharing platforms such as GitHub, no one can use it unless they are explicitly granted permission. This is done by adding a software license, which defines the set of rules and conditions for people who want to use the software."*
Software citation principles	Software license is only mentioned in the use cases table and with an + sign which states: *indicate that the use case would benefit from that metadata if available.*
SSC IG P13 activity	*Ideally licenses should be in rights expression languages*
From FAIR research data toward FAIR and open research software	*Build modular software architectures to allow for reusing parts of research software*
Attributing and Referencing (Research) Software	*(not explicitly discussed)*
Software vs. data in the context of citation	*"Software is a creative work, scientific data are facts or observations In particular, software is generally subject to copyright protection as a creative work that can continue to evolve over time, while scientific data is frequently considered outside the domain of copyright as it is comprised of contextual facts about the world…"*
The science code manifesto	*Copyright: The copyright ownership and license of any released source code must be clearly stated.*
CoSO opportunity note	*Recommendation n° 9: Encourage and facilitate the creation of "legal toolboxes" to ensure the long-term preservation of free software resulting from research.*

FAIR	Relevant	Achievable	Measurable	Benefits
R1.1. (meta)data are released with a clear and accessible data usage license.	***	***	***	***

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

R1.2. (meta)data are associated with their provenance.

Resource	Content
Towards FAIR principles for research software	*Rephrased: "Software metadata include detailed provenance, detail level should be community agreed."* *"Provenance refers to the origin, source and history of software and its metadata. It is recommended to use well-known provenance vocabularies, for instance PROV-O [63], that are FAIR themselves. "*
"5 recommendations for FAIR software"	***Repository:*** *"Using a version control system allows you to easily track changes in your software, both your own changes as well as those made by collaborators."*
Software citation principles	*"...the software metadata recorded as part of data provenance will overlap the metadata recorded as part of software citation for the software that was used in the work. The data recorded for reproducibility should also overlap the metadata recorded as part of software citation. In general, we intend the software citation principles to cover the minimum of what is necessary for software citation for the purpose of software identification."*
SSC IG P13 activity	*"needs metadata that isn't available (dependencies)"*
From FAIR research data toward FAIR and open research software	*Use domain-specific languages for comprehensibility and modularity of research software*
Attributing and Referencing (Research) Software	*(not explicitly discussed)*
Software vs. data in the context of citation	*"...best practices to facilitate reproducibility of computational science involve archiving of the following, in durable, plaintext formats:* *...* *3. descriptive provenance metadata on the software dependencies needed to compile and run the software as well as any input data dependencies"*
The science code manifesto	*(not explicitly discussed)*
CoSO opportunity note	*(not explicitly discussed)*

FAIR	Relevant	Achievable	Measurable	Benefits
R1.2. (meta)data are associated with their provenance.	***	**	*	**

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

R1.3. (meta)data meet domain-relevant community standards.

Resource	Content
Towards FAIR principles for research software	*Rephrased: "Software metadata and documentation meet domain-relevant community standards."* *"we consider aspects of installation instructions (R1.3), software dependencies (I4S), and licensing (R1.1) as part of other principles here, rather than adding another Accessibility principle."*
"5 recommendations for FAIR software"	***Registry:*** *"What metadata does the community registry offer? This is sometimes described in the documentation of the registry, but you can also see for yourself by installing a tool like the OpenLink Structured Data Sniffer. "*
Software citation principles	*"Existing efforts around metadata standards. Producing detailed specifications and recommendations for possible metadata standards to support software citation was not within the scope of this working group. However some discussion on the topic did occur and there was significant interest in the wider community to produce standards for describing research software metadata."*
SSC IG P13 activity	*(not explicitly discussed)*
From FAIR research data toward FAIR and open research software	*"Follow good software engineering practices to achieve high software quality* *Use software virtualization techniques such as Docker to support reusability across platforms* *Use software-as-a-service platforms such as BinderHub for immediate execution* *Use research software observatories for online analytics* *Participate in artifact evaluation processes to evaluate reusability"*
Attributing and Referencing (Research) Software	*(not explicitly discussed)*
Software vs. data in the context of citation	*(not explicitly discussed)*
The science code manifesto	*"There are many well-known open-source licenses: use of a well-known existing license is strongly recommended."*
CoSO opportunity note	*"Recommendation n° 8: Define a common strategy and procedures for evaluating open source software making it sustainable and encouraging technology transfer."*

FAIR	Relevant	Achievable	Measurable	Benefits
R1.3. (meta)data meet domain-relevant community standards	***	**	*	**

Each criteria is evaluated with the following scale:

N/A doesn't appear (white)	* observed in a small subset (one paper)	**medium subset (2-3)	*** large subset (3+ papers)	! disagreeing

Annex C: Infrastructures and existing implementations catering software

C.1 Software archives and institutional repositories

C.1.1 Software Heritage archive

Service Summary

Software Heritage (SWH) is a universal software source code archive.

Aims & Scope:

- SWH archive operates only on software source code artifacts
- The source code and complete history is archived for long-term preservation
- All software artifacts are referenceable with the SWHID intrinsic identifier
- The software is publicly available and accessible online (not in an arctic vault)

URL: https://archive.softwareheritage.org/

Users

Academia, industry, cultural heritage

Purpose

Software Heritage's goal is to collect, preserve and share all software source code publicly available.
SWH is an automatic archive without manual curation of content or metadata

Adoption

The list of repositories which are archived in SWH is long, it includes GitHub, GitLab, Gitorious and more.

Also HAL[44] the national french archives and IPOL[45] journal deposit software in SWH[4647].

Documentation

- **https://www.softwareheritage.org/save-and-reference-research-software/**
- **https://docs.softwareheritage.org/devel/**

Service components

- automatic pull from different forges (GitHub, GitLab, BitBucket),
- intrinsic metadata is extracted from the content itself (not yet visible on the web-app),
- deposited artifacts are accepted only from known sources where metadata was moderated and curated
- Save code mechanism for git, svn and mercurial repositories

Examples (in bold the core intrinsic identifier[43])

- Revision SWHID: **swh:1:rev:0064fbd0ad69de205ea6ec6999f3d3895e9442c2;***origin=https://github.com/rdicosmo/parmap*
- Snapshot SWHID: **swh:1:snp:c7c108084bc0bf3d81436bf980b46e98bd338453**;origin=https://github.com/darktable-org/darktable/
- Code fragment SWHID: **swh:1:cnt:64582b78792cd6c2d67d35da5a11bb80886a6409**;origin=https://github.com/virtualagc/virtualagc;lines=245-261/

[43] https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html
[44] https://hal.archives-ouvertes.fr/
[45] https://www.ipol.im/
[46] https://www.softwareheritage.org/2018/09/28/depositing-scientific-software-into-software-heritage/
[47] https://www.softwareheritage.org/2020/06/11/ipol-and-swh/

C.1.2 Zenodo

Service Summary

Zenodo is an open dissemination research data repository for the preservation and making available of research, educational and informational content.

Aims & Scope:

- Provide a platform for everyone to engage in Open Science
- Allows upload of all data file formats and content type
- Assign persistent identifiers (DOIs) to all records to allow for citability
- Preserve deposited content for the lifetime of the repository

URL: https://zenodo.org/

Users

Academia, industry, cultural heritage, any non-military users

Purpose

Allow everyone to share their outputs openly (up to 50GB per file), exposing provided metadata in standardised formats.

Adoption

More than 1,5 mio records have been shared via zenodo so far, most of them openly accessible.

Documentation

https://about.zenodo.org/

Service components

- automatic integration with GitHub, see https://guides.github.com/activities/citable-code/
- Storage and preservation in CERN Data Centres
- Versioning of records, including concept DOIs
- Curation of records in communities
- Citation recommendations and usage statistics
- API and OAI-PMH for machine exchange

Examples

- *Software: https://doi.org/10.5281/zenodo.4088798*
- Community: https://zenodo.org/communities/fairsfair

C.2 Software journals and publishers

C.2.1 SoftwareX

Service Summary

SoftwareX is an academic journal which focuses specifically on the publication of research software. As per the Aims & Scope:

- The software is given a stamp of scientific relevance, and provided with a peer-reviewed recognition of scientific impact;
- The software developers are given the credits they deserve;
- The software is citable, allowing traditional metrics of scientific excellence to apply;
- The academic career paths of software developers are supported rather than hindered;
- The software is publicly available for inspection, validation, and re-use.

URL: https://www.journals.elsevier.com/softwarex

Users

Academic researchers of all disciplines.

Purpose

The journal aims to disseminate research software, promote re-use of software and provide a mechanism for researchers to receive academic credit (in the sense of a citable DOI) for sharing their code.

Service components

- Publication of a short, descriptive article on Elsevier ScienceDirect's platform (including structured metadata) called an "Original software publication"
- An open source software distribution on the journals' Github space

Examples

- https://doi.org/10.1016/j.softx.2020.100561
- https://doi.org/10.1016/j.softx.2015.06.001

Adoption

According to a ScienceDirect search query, content has grown from 15 publications in 2015 to 100+ this year. (Metrics on readership and software reuse are not readily available)

Documentation

- Journal homepage: https://www.journals.elsevier.com/softwarex

C.2.2 Journal of Open Source Software (JOSS)

Service Summary

The Journal of Open Source Software (JOSS) is a developer friendly, open access journal for research software packages. It is committed to publishing quality research software with zero article processing charges or subscription fees. JOSS JOSS publishes articles about research software, which include a short paper and the software itself, both of which are *openly* peer-reviewed. This definition includes 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.

URL: https://joss.theoj.org

Users

Scholarly software developers who develop software for research in all disciplines.

Purpose

The journal aims to provide an easy-to-use, zero cost to submitters and readers, fully open means for research software developers to get recognition and scholarly credit for their software work, via the academic publishing and citation system.

Service components

- Publication of a short paper on the journal's platform, registered with Crossref, with both the paper text and DOI metadata linking to a deposit of the software in a preservation repository that provides a DOI (e.g., Zenodo) and to the live version of the software (e.g., on GitHub).
- JOSS articles, metadata, and reviews are archived with Portico.

Examples

- https://doi.org/10.21105/joss.01686
- https://doi.org/10.21105/joss.00884

Adoption

JOSS started in May 2016, and published about 100 papers in its first year. As of 16 October 2020, it has published 1053 papers.

Documentation

- Journal homepage: https://joss.theoj.org
- Documentation: https://joss.readthedocs.io/
- Publication Ethics: https://joss.theoj.org/about#ethics

C.3 Software registries / indexers /aggregators

C.3.1 swMath

Service Summary

Provides access to an extensive database of information on actual use (and citation) of mathematical software. Also includes a systematic linking of software packages with relevant mathematical publications. intention is also to offer a list of all publications that refer to a software recorded in swMATH

Aims & Scope:

- Mathematical software registry
- Created almost exclusively from citations in publication
- Linked with zbMATH

URL: https://swmath.org/

Users

Academia, industry, cultural heritage institutions - everyone conducting and supporting research

Purpose

swMath is a software registry for research software in the mathematics domain.

Adoption

Free service, open feedback and possible to contribute

Documentation

Bönisch, S., Brickenstein, M., Greuel, G.-M., & Sperber, W. (2012). SwMATH – citations for your mathematical software. journalId:00006143, 2012. https://doi.org/10.1007/BF03345852

Bönisch, S., Brickenstein, M., Chrapary H., Greuel G.-M., & Sperber W.(2013). "SwMATH – A New Information Service for Mathematical Software." In Intelligent Computer Mathematics, edited by Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, 7961:369–73. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-39320-4_31.

Chrapary, H., Dalitz, W., Neun, W., & Sperber W. (2017). Design, Concepts, and State of the Art of the swMATH Service. Math.Comput.Sci. 11, 469–481. https://doi.org/10.1007/s11786-017-0305-5

Service components

- Search function that searches the following metadata fields: software name, software authors, description, keywords, programming language and MSC classification ("Mathematics Subject Classification")
- Indexed in search engines
- Permanent link to archived code in SWH
- Uses the MSC classification, widely used by mathematical reviewing services and many others to categorize items in the mathematical sciences literature.
- API allows to get the related software, the time chart data or the last 10 publications of each software.
- Software is linked to entities in WikiData

Examples

- **SemiPar software:** https://swmath.org/software/7116
- SageMath software: https://swmath.org/software/825

C.4 Research software training

C.4.1 The Carpentries

Service Summary

The Carpentries aim to be the leading inclusive community teaching data and coding skills. They build global capacity in essential data and computational skills for conducting efficient, open, and reproducible research.
Aims & Scope:

- Training and fostering a community
- Openly licensed, community developed material
- Lessons on software, data use and analysis and for the library community

URL: https://carpentries.org/

Users

Academia, industry, cultural heritage institutions - everyone conducting and supporting research

Purpose

The Carpentries teaches foundational coding, and data science skills to researchers worldwide.

Adoption

Between 2012 and 2019, they have run 2300 workshops in 61 countries and developed 33 official lessons and more in development.

Documentation

An overview of curricula is available at https://carpentries.org/workshops-curricula/

Service components

- Openly licensed training material on basic coding, version control with git, licensing and data analysis using a variety of software
- Instructor training
- Workshops delivering the training material virtually or face to face

Examples

- Workshop webpage: https://softwaresaved.github.io/2020-10-13-ssi-online/
- Lesson example:
  - https://datacarpentry.org/OpenRefine-ecology-lesson/02-filter-exclude-sort/index.html
  - https://swcarpentry.github.io/r-novice-gapminder/

Bibliography

Allen, A., and Schmidt, J. (2015). Looking Before Leaping : Creating a Software Registry. *Journal of Open Research Software*, *3*(e15). http://dx.doi.org/10.5334/jors.bv

Alliez, P., Di Cosmo, R., Guedj, B., Girault, A., Hacid, M.-S., Legrand, A., & Rougier, N. (2019). Attributing and Referencing (Research) Software : Best Practices and Outlook From Inria. *Computing in Science Engineering*, *22*(1), 39‑52. https://doi.org/10.1109/MCSE.2019.2949413

Barnes N., D. Jones, P. Norvig, C. Neylon, R. Pollock, J. Jackson, V. Stodden, and P. Suber. (2011) Science code manifesto. version 1.0. http://sciencecodemanifesto.org. Accessed: July 15th 2020

Ballhausen, M. (2019). Free and open source software licenses explained. Computer, 52(6), 82-86.

Bönisch, S., Brickenstein, M., Greuel, G.-M., & Sperber, W. (2012). SwMATH – citations for your mathematical software. *journalId:00006143*, *2012*. https://doi.org/10.1007/BF03345852

Bönisch, S., Brickenstein, M., Chrapary H., Greuel G.-M., & Sperber W.(2013). "SwMATH – A New Information Service for Mathematical Software." In Intelligent Computer Mathematics, edited by Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, 7961:369–73. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-39320-4_31.

Chrapary, H., Dalitz, W., Neun, W., & Sperber W. (2017). Design, Concepts, and State of the Art of the swMATH Service. *Math.Comput.Sci.* 11, 469–481. https://doi.org/10.1007/s11786-017-0305-5

Clément-Fontaine, Mélanie, Roberto Di Cosmo, Bastien Guerry, Patrick Moreau, and François Pellegrini. (2019). Encouraging a Wider Usage of Software Derived from Research. Research Report. Committee for Open Science's Free Software and Open Source Project Group. https://hal.archives-ouvertes.fr/hal-02545142

Di Cosmo R. (2020) Archiving and Referencing Source Code with Software Heritage. In: Bigatti A., Carette J., Davenport J., Joswig M., de Wolff T. (eds) Mathematical Software – ICMS 2020. ICMS 2020. Lecture Notes in Computer Science, vol 12097. Springer, Cham. https://doi.org/10.1007/978-3-030-52200-1_36

Di Cosmo R., Gruenpeter M., Marmol B., Monteil A., Romary L., Sadowska J. (2020a) Curated Archiving of Research Software Artifacts : lessons learned from the French open archive (HAL). (2020a).**10.2218/ijdc.v15i1.698**

Di Cosmo, R., Gruenpeter, M., & Zacchiroli, S. (2020b). Referencing Source Code Artifacts : A Separate Concern in Software Citation. *Computing in Science & Engineering*. https://doi.org/10.1109/MCSE.2019.2963148 https://hal.archives-ouvertes.fr/hal-02446202

Di Cosmo, R. and Zacchiroli, S. (2017). Software Heritage: Why and How to Preserve Software Source Code. iPRES 2017 - 14th International Conference on Digital Preservation, Sep 2017, Kyoto, Japan. pp.1-10. ⟨hal-01590958⟩

European Commission (2018) Turning FAIR into reality. Final Report and Action Plan from the European Commission Expert Group on FAIR data . Luxembourg Publication Office of the European Union, Luxembourg, 78 pp. https://doi.org/10.2777/1524

FAIRsFAIR .(2020). 'About'. Retrieved from https://www.fairsfair.eu

FAIR Practice TF: Chue Hong, Neil, Cozzini, Stefano, Hoffman-Sommer, Marta, Hooft, Rob, Lembinen, Liisi, ... Teperek, Marta. (2020). Six Recommendations for Implementation of FAIR Practice. European Commission Library doi: 10.2777/986252

Gruenpeter M. and Thornton K. (2018) Pathways for Discovery of Free Software (slide deck from LibrePlanet 2018). https://en.wikipedia.org/wiki/File:Pathways-discovery-free.pdf accessed September 22nd 2020.

Hasselbring, W., Carr, L., Hettrick, S., Packer, H., & Tiropanis, T. (2020). From FAIR research data toward FAIR and open research software, it - Information Technology, 62(1), 39-47. doi: https://doi.org/10.1515/itit-2019-0040

Hervé L'Hours, & Ilona von Stein. (2020). FAIR Ecosystem Components: Vision (Version 02.00). Zenodo. http://doi.org/10.5281/zenodo.3734273

Hinsen, K. (2019). Dealing With Software Collapse, in *Computing in Science & Engineering*, vol. 21, no. 3, pp. 104-108, 1 May-June 2019, doi: 10.1109/MCSE.2019.2900945.

Jones B. M., Boettiger C., Cabunoc Mayes A., Smith, A., Slaughter, P., Niemeyer, K., Gil Y., Fenner M., Nowak, K., Hahnel, M., Coy, L., Allen A., Crosas, M., Sands A., Chue Hong N., Cruse P.,Katz D. S., Goble, C. (2017). CodeMeta: an exchange schema for software metadata. Version 2.0. KNB Data Repository. doi:10.5063/schema/codemeta-2.0

Katz, D. S., Bouquin, D., Hong, N. P. C., Hausman, J., et al. (2019). Software citation implementation challenges. *arXiv preprint arXiv:1905.08674*. https://arxiv.org/abs/1905.08674

Katz DS, Niemeyer KE, Smith AM, Anderson WL, Boettiger C, Hinsen K, Hooft R, Hucka M, Lee A, Löffler F, Pollard T, Rios F. (2016). Software vs. data in the context of citation. PeerJ Preprints 4:e2630v1 https://doi.org/10.7287/peerj.preprints.2630v1

Koers, H., Gruenpeter, M., Herterich, P., Hooft, R., Jones, S., Parland-von Essen, J. & , Staige, C., (2020). Assessment report on 'FAIRness of services'. FAIRsFAIR. https://doi.org/10.5281/zenodo.3688762

Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E., Dominguez Del Angel, V., van de Sandt, S., Ison, J., Martinez, P. A., McQuilton, P., Valencia, A., Harrow, J., Psomopoulos, F., Gelpi, J. L., Chue Hong, N., Goble, C., & Capella-Gutierrez, S. (2019). Towards FAIR principles for research software. Data Science, Preprint, 1–23. https://doi.org/10.3233/DS-190026

Lehväslaiho, H., Parland-von Essen, J., Behnke, C., Laine, H., Riungu-Kalliosaari, L., Le Franc, Y., & Staiger, C.. (2019). D2.1 Report on FAIR requirements for persistence and interoperability 2019 (Version v1.0 Draft). FAIRsFAIR. https://doi.org/10.5281/zenodo.3557381

Research Data Alliance/FORCE11 Software Source Code Identification WG, Allen, A., Bandrowski, A., Chan, P., Cosmo, R. D., Fenner, M., Garcia, L., Gruenpeter, M., Jones, C. M., Katz, D. S., Kunze, J., Schubotz & M., Todorov, I. T. (2020). Use cases and identifier schemes for persistent software source code identification (V1.0). Research Data Alliance. https://doi.org/10.15497/RDA00053

Smith, A. M., Katz, D. S., & Niemeyer, K. E. (2016). Software citation principles. *PeerJ Computer Science*, *2:e86*. https://doi.org/10.7717/peerj-cs.86

Software Heritage blog (2020) Intrinsic and Extrinsic identifiers https://www.softwareheritage.org/2020/07/09/intrinsic-vs-extrinsic-identifiers/

Wilkinson, M., Dumontier, M., Aalbersberg, IJJ., Appleton, G., Axton, M., Baak, A. et al. (2016). The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 3, 160018 (2016). https://doi.org/10.1038/sdata.2016.18