FAIR for Research Software Principles Discussion Document for Community Input

This public document is for community consultation on work carried out by the FAIR4RS Working Group and subgroup activities from July 2020 to February 2021.
We recommend you become a member of the FAIR4RS Working Group before collaborating on this document.

This document examines and discusses a) the output of FAIR4RS Subgroup 1: A fresh look at FAIR for Research Software, b) the output of FAIR4RS Subgroup 4: Review of new research related to FAIR Software, c) the position paper "Towards FAIR Principles for Research software", d) "5 recommendations for FAIR software", e) the output report from FAIR4RS Subgroup 2: FAIR work in other contexts, and e) the definitions of research software produced by FAIR4RS Subgroup 3.

How to contribute
This document will remain open for collaborators to respond to the questions posed or add comments from 24 Feb 2021 until 10 Mar 2021.

Please add your name to the list of collaborators.

If adding new information, please make sure to cite your sources in the References section at the end of the document.

As an online global and diverse community, we expect professional behaviour.
Your contributions are valued by the community. We ask that you help others feel equally valued and welcomed by treating others with the respect and professionalism with which you would like to be treated. Please adhere to the **RDA Code of Conduct**.

Collaborators

Full name	**Your role, organisation, location, discipline, and ORCID** (if you haven't previously provided this info as part of other FAIR4RS WG activities)
Marcos Roberto Tovani-Palone	Researcher, University of São Paulo, Brazil, ORCID 0000-0003-1149-2437
Tom Honeyman	Software Program Manager, Australian Research Data Commons, Sydney, 0000-0001-9448-4023
Joanna Leng	EPSRC funded RSE Fellow, University of Leeds, UK, 0000-0001-9790-162X
Manodeep Sinha	Senior Research Software Scientist, Swinburne University of Technology/ASTRO 3D Centre of Excellence, Hawthorn, VIC 3122. 0000-0002-4845-1228
Sharif Islam	Data Architect, Distributed System for Scientific Collections (DiSSCo). The Netherlands, 0000-0001-8050-0299
Axel Loewe	Assistant Professor, Karlsruhe Institute of Technology (KIT), 0000-0002-2487-4744
Merc Fox	Director, CODATA at UA, University of Arizona, STS, 0000-0002-0726-7301
Udayanto Dwi Atmojo	Postdoctoral Research Fellow, Aalto University, Finland, 0000-0002-6865-0806
Tom Pollard	Technical Director, PhysioNet. 0000-0002-5676-7898
James McNally	Director, NACDA Program on Aging, ICPSR University of Michigan Orcid ID 0000-0002-6807-4538
Malin Sandström	Community Engagement Officer, Working Group project manager, INCF 0000-0002-8464-2494
Ben van Werkhoven	Senior Research Engineer, Netherlands eScience Center, Amsterdam, the Netherlands 0000-0002-7508-3272
Ilian Todorov	Computational Chemistry Lead, UKRI Science and Technology Facilities Council, 0000-0001-7275-1784
Patricia Herterich	Research Data Specialist, Digital Curation Centre, University of Edinburgh, UK 0000-0002-4542-9906
Hugh Shanahan	Professor of Open Science, Department of Computer Science, Royal Holloway, University of London, UK 0000-0003-1374-6015

Mathieu Servillat	Research Engineer, LUTH - Observatoire de Paris, France 0000-0001-5443-4128
Elena Ranguelova	Technology Lead, Netherlands eScience 0000-0002-9834-1756
Catherine Jones	Energy Data Centre Lead, UKRI Science and Technology Facilities Council, 0000-0002-5112-835X
Daniele Tartarini	Research Software Engineer. Department of Computer Science and Insigneo institute of in silico medicine. University of Sheffield, UK 0000-0002-8913-0156

Invitation

Hello all,

After our recent FAIR4RS Townhall, we're pleased to inform you that the first combined output of the FAIR4RS working group is now available. This review document discusses the results of the FAIR4RS subgroups, as well as the paper "Towards FAIR Principles for Research Software (Lamprecht et al. 2020), the 5 recommendations for FAIR software website, and identifies key questions related to defining FAIR for research software. It is now available via Google Docs for community input.

Your feedback on the questions posed in this report will be used to inform the scope, requirements, and priorities for future outputs of the working group, including its final report. Comments on the discussion presented in the report are also welcomed.

We plan to use this public document as the main engagement activity in a two-week period. It will remain open for collaborators to edit from 24 Feb 2021 until 10 Mar 2021.

If you are not available for these two weeks, do not hesitate to get in touch with us, we will continue offering opportunities to provide feedback for future activities/outputs of the group. The next stage of the community process will be the drafting of a revised definition of the FAIR principles for research software, and we will post more details on how to get involved via the FAIR4RS mailing list (RDA FAIR4RS WG posts).

If you require more information, please comment to this post.

Thank you again for your contribution and support,
FAIR4RS WG Steering Committee

Report

February 23, 2021
---DRAFT---

Table of Contents

Introduction

This document is the result of the four subgroups of the FAIR for Research Software working group, which is working under the Research Data Alliance, the Research Software Alliance, and FORCE11. These subgroups independently examined the FAIR principles in relation to software.

FAIR4RS-subgroup1 started with the original FAIR principles (Wilkinson et al. 2016) and worked to

1. Determine what part of the original FAIR principles apply as is to research software;
2. Determine what part of the original FAIR principles doesn't apply at all

to research software; and
3. Determine what part of the original FAIR principles applies to research software, but with a different definition or different details, starting with the original FAIR principles themselves, and not relying on work done by others to apply them to research software, such as by Lamprecht et al. (2020).

This led to a document (Katz et al. 2021) that includes:

- a discussion of the differences between software and data,
- an initial straightforward translation that was collected from the FAIR4RS-subgroup1 participants;
- a discussion about the nuances of the currently defined rules in the context of research software;
- a proposed set of principles adapted to the FAIR research software case;
- a comparison of those proposed principles with the FAIR data principles;
- a set of gaps in our current infrastructure and existing practices that make implementing the proposed principles difficult; and
- a discussion of where the proposed principles fall short of a larger world of fully-open, high-quality, sustainable software developed and maintained by recognized and rewarded people in the context of full-reproducible research.

We refer to these proposed principles as FAIR4RS-subgroup1.

FAIR4RS-subgroup2 looked at the work of FAIR4RS-subgroup1 and provided feedback and comments related to other digital objects that FAIR4RS-subgroup1 did not consider, such as training materials and workflows, to understand how general the FAIR4RS-subgroup1 work was. We refer to this work as FAIR4RS-subgroup2.

FAIR4RS-subgroup3 examined the complexity of defining research software, by gathering definitions of research software and related terms in the literature, and by compiling examples and discussing whether they exemplify research software This was then followed by two workshops with the intention of clarifying the scope of the FAIR principles by identifying for which software artifacts the FAIR principles should be applied. The concept of exclusive and inclusive definitions regarding the usage of the term "Research" were further discussed, as well as further discussion around a small number of examples of research software. This discussion and the preceding compilation work were synthesised as a report portraying a complex landscape of software uses and software examples in research. Furthermore, an analysis of existing definitions resulted in a better

understanding of the complexity of types of software and types of roles software has during the research process. The subgroup identified an important controversy in academia, which is by itself a step forward for the FAIR software roadmap.

FAIR4RS-subgroup4 started with the rewritten FAIR principles for research software (Lamprecht et al. 2020) and worked to:

- identify other work (FAIR4RS WG, 2020) that helped to inform the application of FAIR principles to research software, and examples of software that helped to understand the characteristics of FAIR software;
- discuss and agree how the spirit of the FAIR foundational principles could be interpreted and applied to research software;
- determine which of the rewritten FAIR principles in Lamprecht et al. 2020 applied as written, applied if rewritten, or did not make sense to apply to research software; and
- suggest where further discussion is needed to rewrite, add or delete FAIR guiding principles for research software.

This was undertaken using a survey that sought feedback and reflection on the rewritten FAIR principles in Lamprecht *et al.* 2020. A reading list of other work was compiled which identified potential blindspots, including a lack of attention to relevant work from domains outside of life sciences and physical sciences. The responses to the survey were synthesised to produce a reinterpretation of the FAIR foundational principles for software, as well as identifying common themes and specific criticisms of the Lamprecht et al. 2020 proposed guiding principles for research software. We refer to this work as FAIR4RS-subgroup4.

We additionally consider two other documents/groups that have worked in this space. First, the principles proposed by Lamprecht *et al.* 2020 in "Towards FAIR principles for research software" which we refer to in the remainder of the report as Lamprecht, and second, the recommendations in "5 recommendations for FAIR software," which we refer to as 5RECS.

Then, we compare the different recommendations, ask specific questions, and discuss possible options. We also show two figures that attempt to explain the different aspects of FAIR for Research Software. This document's main goal **and this community consultation period is to get community feedback on these options and the figures.** After this comparison, discussion of options, figures, and references, a detailed table of the recommendations appears as an appendix.

Comparison

Overall, these different recommendations have a number of similarities, as well as some differences.

Crosscutting

The comparison of the work analysed in this report included five crosscutting concerns that require resolution to define a set of FAIR guiding principles for research software.

1. **General vs specific principles**: Most of the questions raised, to some extent, relate to the desired balance of the principles between very general statements and more actionable instructions. General guidance is less tied to specific infrastructure and is thus more long-lasting but is also more difficult to act on without details.
    - How do we balance between principles that are very general and specific, actionable instructions?
2. **Long-term access to software**: All of the recommendations agree that long-term access to the metadata describing the software is important. Archiving of software source code to ensure that software produced from research work is not lost, is seen as crucial in the wider context of research (European Commission, 2020). The definitions of the foundational principles in the various recommendations imply that long-term access to the software itself is also useful to improve its FAIR-ness. However, there are no recommendations for explicitly including this in the guiding principles.
    - Should long-term access to software be considered as a factor for FAIR, and should it be written into the guiding principles?
    - Should long-term access be reserved to source code?
3. **Defining research software**: We may consider two definitions, inclusive and exclusive. Inclusive represents the far end of a spectrum which will include all software that was used, produced, or analyzed in research. An exclusive definition will only consider a small subset of software artifacts that are equivalent in their discovery as reviewed publications (e.g software published on JOSS).
    - Where should the line between the inclusive and exclusive definition be when it comes to applying FAIR principles to

software? Is it realistic or productive to require “all software” in research to be FAIR?

4. **Defining software**: There is also an overall question about different types of objects and instances to which the FAIR principles for research software should apply. This is, in part, because research objects are related, as discussed in the next point, but also because software is a fuzzy concept that could be applied to source code in a variety of languages, executables, scripts, workflows, or even input files that control how a system operates. Here we propose to define software itself as "A set of instructions[1] that performs some action, either as source code (machine- and human-readable) or executable." This definition includes scripts and workflows, but does not include input files, documentation, data, infrastructures, or services.
    - Is this definition of software, used to define the set of objects to which the FAIR principles for research software should apply, reasonable in this context?
5. **FAIRness of related research objects**: A major difference in the way that the recommendations approach the definition of FAIR guiding principles for research software is how each considers related objects including software dependencies, references to required data objects, and documentation. This includes concepts such as whether FAIR is recursive, i.e. a digital research object is only “fully FAIR” if the objects it builds on are also FAIR.
    - Should the FAIR guiding principles for software include recommendations that related digital research objects which are required to understand or execute the software, such as software and data dependencies, are also FAIR?

Findable

Regarding "Findable," all recommendations agree that this is a good principle. All agree that for software to be findable, it should be identifiable, that software should be defined with metadata associated with the software, that an identifier should be part of this metadata, and that this metadata should be available and searchable through some type of a resource.

However, there are also some differences.

One is related to granularity. FAIR4RS-subgroup1 discusses ten levels of

1The instructions should be capable of general expression (Turing complete), but many instances of software will be limited to performing specific actions.

granularity at which software can be identified, FAIR4RS-subgroup4 and Lamprecht work at the level of software versions and software packages, and 5RECS only discusses packages. FAIR4RS-subgroup2 considers versions at the level of snapshots and releases.

Additionally, FAIR4RS-subgroup4 brings in the idea of identifiers being compatible with best practices in software engineering such as respecting semantic versioning and automated generation of artefacts, i.e. applying the FAIR principles should not hinder the ability to use automated builds and continuous integration systems which may generate metadata and identifiers, while none of the other recommendations get to this level of detail.

FAIR4RS-subgroup4 also goes into more detail about the metadata that should be associated with the software, and the challenges of defining “rich” metadata, compared to the other recommendations.

Finally, while the original FAIR principles discuss where metadata are stored very generally ("F4. (Meta)data are registered or indexed in a searchable resource"), and Lamprecht basically agrees with this, 5RECS suggests "a community repository", FAIR4RS-subgroup1 brings up that there is a gap in practices between various registries and repositories, including Software Heritage. FAIR4RS-subgroup4 again goes into more detail about different types of registries and what can be considered to be a registry.

Findable options

1. Should the FAIR principles for research software discuss the levels of granularity identifiers should be assigned?
    a. If so, how many and which levels should be discussed?
2. Are the FAIR principles for research software, involving identifiers and metadata, compatible with best practices in software engineering around the management and versioning of artefacts?
3. Should the FAIR principles for research software discuss what metadata should be provided for software, or what standards the metadata should follow?
4. How much should the FAIR principles for research software discuss the current state of where metadata associated with software can be stored and searched? Should it make specific recommendations?

Accessible

The general idea of software accessibility as a foundational principle is again agreed upon by all recommendations, but again with differences. These differences include what accessibility means (readability, executability, removal of barriers to use), what exactly "software" means (a version, source code, an executable), if coding standards and practices need to be followed, if dependencies also need to be equally accessible, etc. In general, FAIR4RS-subgroup1 is the most general, while the other recommendations add details and limits, ranging from 5RECS to Lamprecht to FAIR4RS-subgroup4.

In terms of how the software is retrieved, 5RECS doesn't discuss this, while FAIR4RS-subgroup1 points out implementation challenges, including the potential role of package managers, version control systems, and how commercial software is treated. Lamprecht says that this can be achieved by using a repository or registry. FAIR4RS-subgroup4 agrees with FAIR4RS-subgroup1 on the role of package managers and version control, and also notes that, unlike data, most software is already retrieved through well-accepted protocols, such as https or ftp/sftp/scp.

Another difference is the role of authentication and authorization in accessing software, which Lamprecht and FAIR4RS-subgroup1 agree with, 5RECS ignores, and FAIR4RS-subgroup4 mostly agrees with but some questioned if this could be interpreted in the same way for software as it is for data.

FAIR4RS-subgroup 4 suggests that accessibility should include ensuring that barriers to use (including physical, social, or technological barriers) are addressed, the usage of the term in other areas of software engineering, though this could be considered part of reusability. FAIR4RS-subgroup 2 notes that the terminology is confusing and may mean the principle is not well understood across domains, if the definition is strictly around protocols for access.

The recommendations generally agree with metadata being accessible even when the software is no longer available.

Accessibility options

1. Should the FAIR principles for research software discuss the details of

how software is accessed?
    a. Is this just http/https? And is it the same as for any other research object?
    b. Or should package managers be discussed, which may internally use http/https but wrap this with a higher-level of access?
2. Should the FAIR principles for research software discuss at what granularity software is accessed?
3. Can the FAIR principles for research software apply to commercial (closed source) software?
4. Can accessing FAIR software require authentication and authorization?
5. Should the FAIR principles for research software ensure that barriers to use (including physical, social, or technological barriers) are addressed?

Interoperable

The FAIR data principles describe interoperable as "The data usually need to be integrated with other data. In addition, the data need to interoperate with applications or workflows for analysis, storage, and processing." Lamprecht interprets this as:

1. A set of independent but interoperable objects interoperate to produce a runnable version of the software, including libraries, software source code, APIs and data formats, and any other resources for facilitating that task.
2. A stack of digital objects interoperate to execute a given task. The stack includes the software itself, its dependencies, other indirect dependencies, the whole execution environment including runtime dependencies and the operating system, the execution environment, dependencies, and the software itself.
3. Workflows, which interconnect different standalone software tools that interoperate to transform one or more data sets into one or more output data sets through agreed protocols and standards.

5RECS doesn't specifically address interoperability in the same way. FAIR4RS-subgroup1 limits its definition of interoperability to the exchange of data or metadata between software, which roughly corresponds to Lamprecht's points 1 and 3, and believes that the sense of building software and then executing it in an environment (Lamprecht's point 2) is not interoperability but rather usability (under reusability in the FAIR principles). FAIR4RS-subgroup 4 more or less aligns with FAIR4RS-subgroup 1 on Lamprecht's points 1 and 3, but is unsure about point 2. FAIR4RS-subgroup

2 expands on the role of workflows and workflow management systems.

There is also some feeling that interoperable might include the use of controlled vocabularies for the metadata about software in repositories in Lamprecht, and that it includes recording of metadata using standards such as CodeMeta and the Citation File Format in 5RECS. FAIR4RS-subgroup 1 doesn't think this is part of the FAIR principles for research software, as it is already covered by the FAIR data principles' discussion of metadata, and FAIR4RS-subgroup 4 is uncertain, with some members agreeing with Lamprecht, and some suggesting that because all software is written in a formal language, there is inherent standardisation and machine readability, making this FAIR principle redundant for software.

Lamprecht also considers the need for controlled vocabularies for the data consumed and produced by software, which 5RECS doesn't consider. FAIR4RS-subgroup 4 agrees with Lamprecht, while FAIR4RS-subgroup 1 requests qualified references to such objects, which are covered by the original FAIR principles' discussion of metadata. FAIR4RS-subgroup 2 agrees with FAIR4RS-subgroup 1, noting that this also works for workflows and scripts and all objects where the process is explicit as opposed to being buried in the code.

Interoperable options

1. Is the process of building software (including determining and accessing dependencies) and running it in a given environment part of FAIR principles for research software?
    a. If so, is it part of interoperability or reusability?
2. Should the FAIR principles for research software explicitly include requirements on the metadata used to describe software, or is this already covered in the FAIR data principles?
    a. If already covered, should it explicitly insist that metadata is FAIR?
3. Similarly, should the FAIR principles for research software explicitly include requirements on the data consumed and produced by software, or is this already covered in the FAIR data principles?
    a. If already covered, should it explicitly insist that data is FAIR?

Reusable

The FAIR data principles state that their "ultimate goal is to optimise the

reuse of data. To achieve this, metadata and data should be well-described so that they can be replicated and/or combined in different settings." FAIR4RS-subgroup 1 considers "optimise" to be too strong a statement, suggesting "enable and encourage" instead.

As Lamprecht notes: "Reusability in the context of software has many dimensions". For software, consideration needs to be given about whether reuse simply means optimising the reuse of data or, also, the reuse of software.

Lamprecht takes the view that "at its core, reusability aims for someone to be able to reuse software reproducibly" and describes four scenarios:

1. reproducing the same outputs reported by the research supported by the software,
2. (re)using the code with data other than the test one provided to obtain compatible outputs
3. (re)using the software for additional cases other than those stated as supported, or
4. (iv) extending the software in order to add to its functionality.
5. Relevant,possible 5th step of reusability; reimplementation, code well written, well structured and well documented enough that it can be understood and be rewritten in another language/for another computational platform so that its overall ideas and modes of implementation can be reused. This happens a lot in (neuro)modelling, where the original model requires a simulator or language the modeller doesn't have access to or cannot run for other reasons.

5RECS does not explicitly define reusability but suggests that is associated with public accessibility of source code, collaboration, and reproducibility of results.

FAIR4RS-subgroup 1, FAIR4RS-subgroup 2, and FAIR4RS-subgroup 4 agree that it is important to recognise that software is dependent on other software, and software should be structured to maximise its potential use or reuse, following software best practice such as encapsulation, or recording of dependencies. However FAIR4RS-subgroup 2 queries the application of encapsulation, is software that calls a service or API not reusable? This leads to a new reusability principle from FAIR4RS-subgroup 1 that "Software includes qualified references to other software" and that, to be FAIR, external data objects required to execute the software must be FAIR as well. This aligns with the discussion from FAIR4RS-subgroup 4 on the interpretation of the FAIR foundational principles; however they go further

and suggest that the FAIR-ness of a piece of software is increased when both the data *and* the software referenced by it are also made FAIR-er.

FAIR4RS-subgroup 4 also goes into more detail, suggesting that for software to be reusable, it should also be maintainable (which Lamprecht also emphasises) and dependable (able to be built on for other purposes). This latter is encapsulated in additional principles from FAIR4RS-subgroup 4 which align with the vision of, but was considered out of scope for, FAIR4RS-subgroup 1.

Both FAIR4RS-subgroup 1 and FAIR4RS-subgroup 4 take a much wider, and similar, view of reuse than Lamprecht. FAIR4RS-subgroup 1 suggests it should cover "replicated, combined, reinterpreted, reimplemented, and/or used" and FAIR4RS-subgroup 4 suggest it should be usable, extensible, integratable, maintainable, well-documented and reproducible.

FAIR4RS-subgroup 1 does not consider "executability" to be a necessary feature for software to be FAIR. However, FAIR4RS-subgroup 4 implies that usability is important for reusability. This could be seen as at odds, but may also point to requiring different interpretations of usability for different types of software (source code can be read, built into executables, used in libraries, etc., while executables can be incorporated into other software or run, etc.) or maturity level. FAIR4RS-subgroup 2 agree that reuse through inspection is critical and more sustainable than reuse through execution.

There is an agreement from all five efforts (Lamprecht, 5RECS, FAIR4RS-subgroup 1, FAIR4RS-subgroup 2, FAIR4RS-subgroup 4) that a clear license is an essential principle for FAIR software. FAIR4RS-subgroup 4 additionally considered that whilst an open source license was not required for software to be FAIR, it helped make software FAIR. There is also agreement that "software is associated with detailed provenance" if we consider version control systems to capture that information.

Finally, whilst there is agreement that software should be described with a plurality of accurate and relevant attributes, it was noted by both FAIR4RS-subgroup 4 and FAIR4RS-subgroup 2 that care must be taken with the way "community standards" are interpreted to take into account the constant evolution of standards, and the cross-domain and community nature of software.

Reusable options

1. Does the principle "Software meets domain-relevant community standards" need more explicit detail?
    a. If so, should documentation be included?
    b. If so, should usability be defined?
    c. If so, should it also consider the type of software and the maturity level?
2. Should a new principle be added so that "Software includes qualified references to other software"?
    a. If so, does this imply that any references to external data objects required to execute the software should be fully qualified and the data be FAIR as well?
    b. If so, does the software that is referenced need to be FAIR as well?
3. Should a new principle be added so that "Software is dependable and can be built on by other software and research"?
    a. If so, should this make explicit what is expected for the software to be dependable, or should this remain at a high-level, or defined by community norms?

Figures

Two potential figures were created (or adapted) as part of the work of subgroup 1, as follows.

Your comments on these figures, including what is confusing, what is missing, etc. are welcome.

Are one or both of these figures useful in explaining FAIR for research software?

Could they be combined? If so, how?

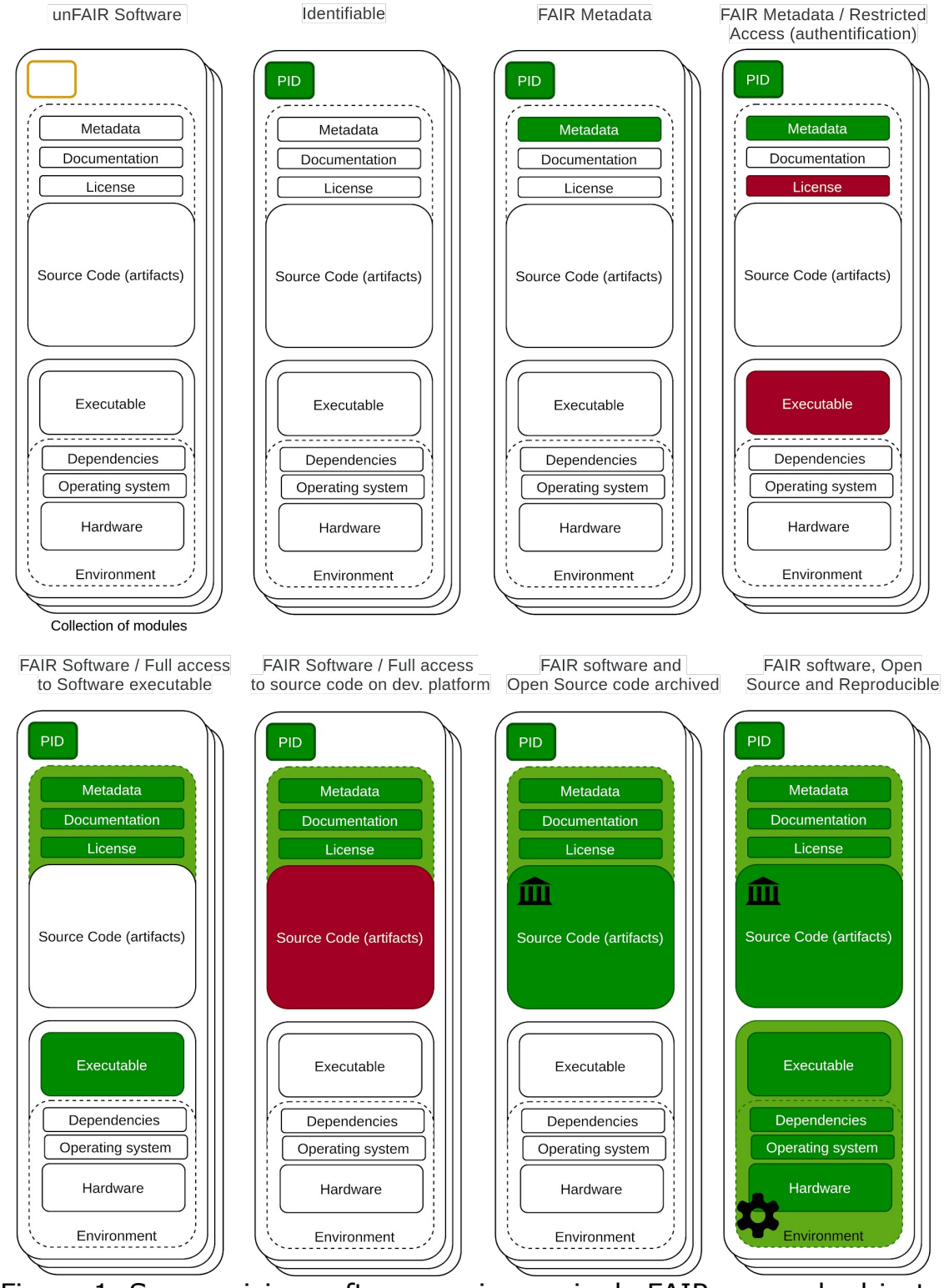

Figure 1. Summarizing software as increasingly FAIR research objects

(Credit: Morane Gruenpeter, inspired by the FORCE11 diagram[2])

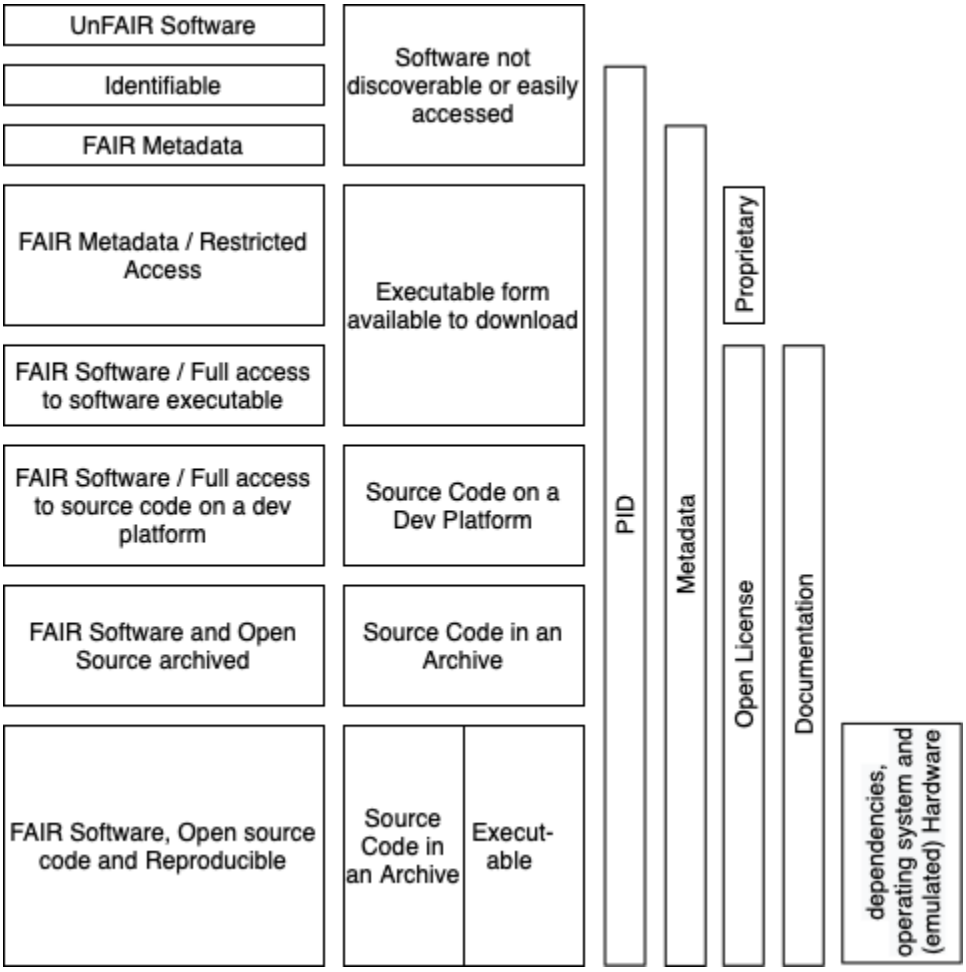

Figure 2. Summarizing software as increasingly FAIR research objects. Left column: labels for different levels of FAIR+. Middle column: software artifact. Right column: Accompanying objects and information. (Credit: Tom Honeyman)

2 https://www.force11.org/fairprinciples

References

European Commission. Directorate General for Research and Innovation. (2020). Scholarly infrastructures for research software: report from the EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS. Publications Office. https://doi.org/10.2777/28598

FAIR4RS WG. (2021). FAIR4RS Subgroup 4 - reading list of new research (Version 1.0) [Data set]. Zenodo. http://doi.org/10.5281/zenodo.4555865

Gruenpeter, M., Di Cosmo, R., Koers, H., Herterich, P., Hooft, R., Parland-von Essen, J., Tana, J., Aalto T. Jones, S. (2020). M2.15 Assessment report on 'FAIRness of software' (Version 1.1). Zenodo. https://doi.org/10.5281/zenodo.4095092

Katz, D.S., Gruenpeter, M., Honeyman, T., Hwang, L., Wilkinson, M.D., Sochat, V., Anzt, H., Goble, C. and FAIR4RS subgroup 1 (2021). A Fresh Look at FAIR for Research Software. arXiv: 2101.10883.

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. Ll., Chue Hong, N., Goble, C., & Capella-Gutierrez, S. (2020). Towards FAIR principles for research software. Data Science, 3(1), 37–59. https://doi.org/10.3233/DS-190026.

Netherlands eScience Center / DANS (n.d.), Five Recommendations for FAIR software. Available online at: https://fair-software.nl/.

Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., … Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data, 3(1). https://doi.org/10.1038/sdata.2016.18.

Appendix A: Analysis of software guidelines

This appendix uses annexe B of the FAIRsFAIR assessment report on 'FAIRness of software' (Gruenpeter et al., 2020) as a starting point, and then adds summaries and discussions of FAIR4RS-subgroup 1 and FAIR4RS-subgroup 4. It provides additional background material to inform the report and options presented in it.

F. Findable

The first step in (re)using data is to find them. Metadata and data should be easy to find for both humans and computers. Machine-readable metadata are essential for automatic discovery of datasets and services, so this is an essential component of the FAIRification process.

Resource	**Content**
Towards FAIR principles for research software	*Findability is a fundamental principle, since it is necessary to find a resource before any other consideration. The main concern of findability for research software is to ensure software can be identified unambiguously when looking for it using common search strategies. Such strategies include the use of keywords in general-purpose search engines like Google, as well as specialised registries (websites hosting software metadata) and repositories (websites hosting software source code and binaries). Findability can be improved by registering the software in a relevant registry, along with the provision of appropriate metadata, providing contextual information about the software. Registries typically render metadata in a web-findable way and can provide a DOI. Some registries and repositories allow annotating software using domain-agnostic or domain-specific controlled vocabularies, increasing findability via search engines further. In the following we discuss how the original four Findability principles apply to the findability of research software.*
"5 recommendations for FAIR software"	*Register your code in a community registry - WHY THIS IS IMPORTANT* *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. Community registries typically employ metadata to describe each software package. 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.*
FAIR4RS-subgroup1	*F. The first step in (re)using software is to find it. Metadata and software should be easy to find for both humans and computers. Machine-readable metadata are essential for automatic discovery of software, so this is an essential component of the FAIRification process.* We believe that findable is an important foundational principle for

	software. We also suggest removing the reference to “services.” While software is definitely a component of any service (and a component that should be FAIR), services are considered here an instantiation of software, not the software itself. Services present an additional series of challenges which we have not considered here.
FAIR4RS-subgroup2	*Workflow findability is foundational - we even have registries dedicated to workflows. What is a workflow wrt software or service is interesting. A workflow can be:* ● *A specification in a WfMS specific or common language (e.g. CWL) with test or exemplar data;* ● *+ an implementation of that design in a WfMS;* ● *+ an instantiation of that implementation ready to be run with input data and parameters set and computational services / containers; - this is not the same as the “instantiation of software” as above I suspect. It's more the configuration of the workflow.* ● *+ a run result with intermediate and final data products and provenance logs.* *Training materials related to the software should also be findable.*
FAIR4RS-subgroup4	*Findable software should:* ● *Include identifiers which enable location of a specific version* ● *Be catalogued in a registry or package manager* ● *Be linked to related research objects, including previous versions [note this links to Subgroup 1’s proposed I2/I3]* ● *Have machine-readable metadata that enables search engines and discovery across different categories (e.g. features, domain, programming language, author)* *Specific clarifications in response to “Towards FAIR Principles...”* ● *The narrow wording of “software” excludes objects on the boundary of software.* ● *Much of what might be considered “Findability” for software has been addressed by package managers* ● *Metadata should specify how software can be translated between its written and its executable state* ● *Machine-readable metadata must make all direct and indirect dependencies findable, using version-specific identifiers.* ● *Metadata describing software have to follow a commonly agreed upon standard.*

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

Resource	Content
Applicability of principle to FAIR for Research Software	Direct application (use * system? **** - highly applicable * -not at all)
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."*
FAIR4RS-subgroup1	*F1. Software is assigned a globally unique and persistent identifier* This guiding principle is fundamental for any research output, but note that it can take some extra effort from the software creators today to acquire a global and persistent identifier. In Section 2, we noted several differences for software development and publishing, both in terms of current practices and in the functionality and existence of relevant infrastructure that might achieve this aim. The creators can use an archive or an institutional repository to keep software and acquire a persistent identifier for their software. However, the identification target might be difficult to choose. As presented in Figure 1 (from Research Data Alliance/ FORCE11 Software Source Code Identification WG et al., 2020), an identification target can be at one of many different granularity levels that are found in a complete software project. For reproducibility for example, it is important to identify a specific version, which means that identifying the full project isn't specific enough. Furthermore there is still a lack of community agreement when it comes to identifying software; see Gaps 1, 2 and 4 in Section 5.
FAIR4RS-subgroup4	*All believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of "Towards FAIR Principles...":* ● *Identifiers should not be restricted to releases. Every version, release or not, should ideally be citable.* ○ *Suggest rewording to "Software and its associated metadata are assigned a global, unique and persistent identifier." [compatible with Subgroup 1]* ● *Recognise that the use of identifiers should be compatible with best practice in software engineering such as respecting semantic*

	versioning and automated generation of artefacts. [compatible with Subgroup 1 but may require additional consideration]

F2. data are described with rich metadata

Resource	**Content**
Applicability of principle to FAIR for Research Software	Direct application
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.*"
FAIR4RS-subgroup1	*F2. Software is described with rich metadata (defined first by R1 below, and then by the original FAIR principles for metadata)* This guiding principle is reasonable and important when it comes to understanding what the software can do and where it comes from. However, the extent and completeness of the metadata is not yet agreed upon by the research community; see Gaps 1 and 3 in Section 5. As noted in Section 2, software structure can be complex, which adds complexity with the metadata (see Gap 5) and with documentation, which might be considered a metadata element (see Gap 6). As discussed above, there are several relevant guiding principles that apply without alteration to metadata for digital objects, including software. In order to capture this, we propose changing the wording for this principle to: "Software is described with rich metadata (defined first by R1b below, and then by the original FAIR principles for metadata)"

	The specific principles are F1, F4, A1, A1.1, A1.2, I1, I2, I3, R1, R1.1, R1.2, and R1.3.
FAIR4RS-subgroup2	*As an example, a profile like Workflow-RO-Crate sets out to define (i) what is expected to be packaged with a workflow (incl Data) and (ii) metadata about it (using schema.org) and (iii) how it is described as steps (e.g. CWL). This adheres to workflow is described with rich metadata.*
FAIR4RS-subgroup4	*Most people believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of "Towards FAIR Principles…":* ● *It isn't yet clear what "rich metadata" means in the context of software, and this should be elaborated. GO-FAIR suggests that "Rich metadata implies that you should not presume that you know who will want to use your data, or for what purpose. So, as a rule of thumb, you should never say 'this metadata isn't useful'; be generous and provide it anyway!" but it is unclear if there are any issues in practice for software. [Probably compatible with Subgroup 1, but R1.3 may not directly address this]* ● *A way of stating the metadata standards is required, if machine processing is to be enabled. [Probably compatible with Subgroup 1, but R1.3 may not directly address this]*

F3. metadata specify the data identifier

Resource	Content
Applicability of principle to FAIR for Research Software	Not obvious
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)*
FAIR4RS-subgroup1	*F3. Metadata clearly and explicitly include the identifier of the software they describe* This guiding principle is reasonable. However, there can be many identifiers to different artifacts that are under the same software project; see Gaps 4 and 5 in Section 5.
FAIR4RS-subgroup2	*Workflows behave like data here - each workflow may have an identifier. The components of a workflow may also have identifiers, this appears to be analogous to the "project"?*
FAIR4RS-subgroup4	*Most people believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of "Towards FAIR Principles..." rewritten version:* ● *May not be useful - enough to have references for previous and next versions. [compatible with subgroup 1]* ● *It would be infeasible to request rich metadata for all old versions, some of which might not be runnable anymore. This is linked to the challenge of understanding what rich metadata is for software. If we assume it includes information generated at compile time, the wording in the rewritten principle in the paper is problematic. [compatible with subgroup 1]*

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

Resource	Content
Applicability of principle to FAIR for Research Software	Direct application
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. "*
FAIR4RS-subgroup1	*F4. Software is registered or indexed in a searchable resource* This guiding principle is reasonable. However, registering software is a complex subject. Current common practice in registries is to identify the software project (see swMath, ASCL or Wikidata) rather than specific software outputs, and this will present a challenge for adopting FAIR software principles; see Gaps 1, 2 and 4 in Section 5. Also see the software structure complexity gap (Gap 5), related to identifiers for different parts of the software.
FAIR4RS-subgroup4	*Most people believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of “Towards FAIR Principles...” rewritten version:* ● *The term and function of a “software registry” is not well defined, so the application of the principle is unclear. [compatible with subgroup 1]* ● *Unclear that there needs to be a specific requirement for registries, principle should be generalised to support other mechanisms for searching for software, by ensuring metadata follows appropriate standards. [probably compatible with subgroup 1]* ● *Unclear that it has to be a specific software registry, rather than a general research object registry. [compatible with subgroup 1[* ● *Code repositories could be classed as “searchable software repositories” [probably compatible with subgroup 1]*

A. Accessible

Once the user finds the required data, she/he needs to know how can they be accessed, possibly including authentication and authorisation.

Resource	Content
Towards FAIR principles for research software	*In the original FAIR Guiding Principles, accessibility translates into retrievability through a standardized communication protocol (A1) and accessibility of metadata even when the original resource is no longer accessible (A2). These principles clearly also apply to software. Interpreting accessibility also as the ability to actually use the software (access its functionality), however, we found mere retrievability not enough. In order for anyone to use any research software, a working version of the software needs to be available. This is different from just archiving source code, even in comprehensive and long-term collections like the Software Heritage archive. To use software, a working version (binary or code) has to be either downloadable and/or accessible e.g., via a web interface, along with the required documentation and licensing information. Accessibility requirements depend on the software type, e.g., web-applications, command-line tools, etc. For example, software containers allow the use across different operating systems and environments, e.g., local computers, remote servers, and high-performance computing (HPC) installations. Cloud-based servers can execute existing pieces of code as a service, as software made available through a web interface or via Jupyter Notebooks [44]. Notebooks allow others to see the results and the narrative alongside the code used to generate them.* *Furthermore, even for software that can be downloaded or accessed without restrictions, being able to run it might also depend on, for example, data samples, (paid) registration, other (proprietary) software packages, or a non-free operating system like Windows or macOS. For data, the FAIR principles demand that "(Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation" (I1) and in that sense discourage the use of proprietary data formats. This is in our view, however, different from transparent dependencies for running software.* *It is worth to re-emphasize that research software are not single, isolated, digital objects. As further discussed for Interoperability, research software interoperate at different levels with other digital objects including other software, and might have different available versions and/or web-based deployments. Still, all implementations should be considered as part of a single entity for the considerations on accessibility with metadata, as to ensure appropriate links among them (see F1, F3). Since accessibility, interoperability and (re)usability are intrinsically connected for research software, 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 recommendationsf or FAIR software"	*Use a publicly accessible repository with version control - WHY THIS IS IMPORTANT* *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. Taken together, this ensures that your software has the best chance of being used by as many people as possible while promoting transparency.
FAIR4RS-subgroup1	*A. Once the user finds the required software, they need to know how it can be accessed, possibly including authentication and authorization.* We believe that accessible is an important foundational principle for software.
FAIR4RS-subgroup4	*Accessible software should:* ● *Be retrievable through a resolvable identifier, using a standard protocol e.g. https* ● *Be able to be inspected and/or executed; as part of this it should include sufficient documentation* ● *Use open metadata* ● *Follow good practice in software accessibility, i.e. making it possible for those with impairments to use the software. These include, but are not limited to, physical, social and technological barriers.* ● *Follow relevant coding standards and good practice* ● *Be accessible in the long-term (but this needs to be reconciled with making all versions identifiable)* *Also, to be accessible, any dependencies required by the software should also be fair, and available via the same protocol.*

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 recommendationsf or FAIR software"	*(not explicitly discussed)*
FAIR4RS-subgroup1	*A1. Software is retrievable by its identifier using a standardised communications protocol* This guiding principle is reasonable in the abstract, but unclear how to implement it for different types of software, particularly for commercial software. In general, open source software is retrievable by its identifier using a package manager, version control, or similar programmatic download service.
FAIR4RS-subgroup2	*The same for workflows and training materials.* *However, this principle is not necessarily well understood across domains. With respect to training materials, the term "accessibility" (protocols here) can be confused with accessibility in terms of support for people with some impairment.*
FAIR4RS-subgroup4	*Most people believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of "Towards FAIR Principles..." rewritten version:* ● *Better expressed as "(Meta)data and code are accessible by their identifier using a standardized communications protocol" [Compatible with Subgroup 1]* ● *Less requirement for A1 and A2, as there is better agreement on standard protocols for accessing software, e.g. HTTP(S) [Compatible with Subgroup 1, but consideration should be given to rephrasing]* ● *Are protocols for sharing metadata compatible with the way that programming languages exchange information? [May require discussion - related to the rich metadata issue]*

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

Resource	Content
Applicability of principle to FAIR for Research Software	Not obvious, though may be because the protocols are widely implemented.
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. "*
FAIR4RS-subgroup1	*A1.1 The protocol is open, free, and universally implementable* This guiding principle is reasonable in the abstract, but it is unclear how to implement it for different types of software, particularly for commercial software.
FAIR4RS-subgroup2	*Does the protocol need to support all possible formats? Perhaps the principle states that to be FAIR we should use an open protocol to download software or workflows. It would be analogous to data.*
FAIR4RS-subgroup4	*Most people believed this principle applied / applied with rewriting, but some felt it did not apply to software, and it was unclear what it meant in a software context. [Compatible with Subgroup 1].*

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	*(not explicitly discussed)*

recommendations for FAIR software”	
FAIR4RS-subgroup1	*A1.2 The protocol allows for an authentication and authorisation procedure, where necessary* This guiding principle is reasonable.
FAIR4RS-subgroup2	*(not explicitly discussed)*
FAIR4RS-subgroup4	*Most people believed this principle applied / applied with rewriting, but some felt it did not apply to software, and it was unclear what it meant in a software context. [Compatible with Subgroup 1].*

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

Resource	Content
Applicability of principle to FAIR for Research Software	Clear examples of applicability.
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)*
FAIR4RS-subgroup1	*A2. Metadata are accessible, even when the software is no longer available* This guiding principle is reasonable, and some mechanisms for achieving this already exist and are in use for some research software already. For instance, software metadata can be captured in domain specific registries like swMath.org or the Astrophysics Source Code Library (ASCL), in general repository solutions like Zenodo, or via a persistent identifier scheme like DOIs.
FAIR4RS-subgroup2	*Also applies to workflows, where this would be equivalent to registering them in long term registries such as workflowhub.eu*
FAIR4RS-subgroup4	*Most people believed this principle applied, but some felt it did not apply to software in isolation from other research objects / metadata. [Compatible with Subgroup 1].*

I. Interoperable

The data usually need to be integrated with other data. In addition, the data need to interoperate with applications or workflows for analysis, storage, and processing.

Resource	Content
Towards FAIR principles for research software	*The IEEE Standard Glossary of Software Engineering Terminology [46] defines interoperability as the "ability of two or more systems or components to exchange information and to use the information that has been exchanged". This definition is further complemented by semantic interoperability, ensuring "that these exchanges make sense – that the requester and the provider have a common understanding of the 'meanings' of the requested services and data." [47]. When examining the FAIR data principles from a research software perspective, interoperability turns out to be the most challenging among the four high-level principles. This is not surprising given the complexity of the software interoperability challenges that form a research area of its own [48–52].* *Already for data and its associated metadata, interoperability has been found to be "the most challenging of the four FAIR principles. This, in part, is due to interoperability not being well understood" [53]. In contrast to the rather static nature of data, research software are live digital objects that interact at different levels with other objects, e.g., other software, managed data, execution environments; and either directly and/or indirectly, as scripts or as part of a workflow (see Fig. 1). The interoperability principles are therefore even more challenging to apply to software, some are not directly applicable, others need to be rephrased and even new principles need to be defined to appropriately address the dynamic nature of software.* *Software interoperability can be defined from three different angles:* *1. for a set of independent but interoperable objects to produce a runnable version of the software, including libraries, software source code, APIs and data formats, and any other resources for facilitating that task;* *2. for a stack of digital objects that should work together for being able to execute a given task including the software itself, its dependencies, other indirect dependencies, the whole execution environment including runtime dependencies and the operating system, the execution environment, dependencies, and the software itself; and* *3. for workflows, which interconnect different standalone software tools for transforming one or more data sets into one or more output data sets through agreed protocols and standards.* *Thus, interoperability for software can be considered both for individual objects, which are the final product of a digital stack, and as part of broader digital ecosystems, which includes complex processes and workflows as well as their interaction [6,54,55]. Different pieces of software can also work together independent of programming languages, operating systems and specific hardware requirements through the use of APIs and/or other communication protocols.* *Software metadata isa necessity for interoperability. They provide the context in which the software is used and contributes towards provenance, reproducibility and reusability. However, a balance is needed between the detail level and its generation cost. Depending on whether research software is considered as an individual product or as part of an ecosystem, the associated metadata might differ [28,56,57], with workflows having specific mechanisms to capture it through their specifications, e.g., using Common Workflow Language (CWL) [58,59] and/or Workflow Description Language (WDL) [60], among others. This metadata should include software version, dependencies (including which version), input and output*

	data types and formats (preferably using a controlled vocabulary), communication interfaces (specified using standards like OpenAPI), and/or deployment options. *Another aspect associated with interoperability is the ability to run the software in different operating systems, i.e. software portability. Software portability strongly depends on the availability of the full execution stack in other operating systems (vertical axis in Fig. 1), which may not always be given. This dependency on other digital objects to have a working software is further extended in the newly introduced FAIR principle I4S. The present tendency to package software and its dependencies, in software containers e.g., Docker, Singularity, Rocket, contributes to enhanced software portability. Although these differences are not negligible, given that these terms are often used interchangeably, we will be considering both under the FAIR principle of interoperability, highlighting any issues that arise due to this divergence.*
"5 recommendations for FAIR software"	*(not directly addressed)*
FAIR4RS-subgroup1	*I. The software usually needs to communicate with other software via exchanged data (or possibly its metadata). Software tools can interoperate via common support for the data they exchange.* Interoperation between data expresses a reciprocal or concomitant relation. Two data sources can be said to interoperate if they can, with relative ease, be integrated in a way that forms a uniform third object. They are equal contributors to the end result. The potential for integration is commonly taken to be good practice in software engineering, but the nature of that relationship is different. There is a contrast between direct or asymmetrical, and indirect or symmetrical integration. First, there is the direct and asymmetrical integration between a piece of software and its dependencies. As implied by the label, the software becomes dependent on the availability and robustness of those dependencies. The dependencies are integrated into the primary software object. This sense of integration does not seem to reflect the reciprocal relation expressed by "interoperability." Second, there is an indirect and often symmetrical integration between independent software objects that can or do exchange data. This could be in the form of information passed between two running instances of software (e.g., services), or it could be in the form of support for common data formats read or written by both software packages. This sense of integration does reflect the reciprocal relations expressed by interoperability. We propose that this foundational principle focus on a sense of interoperability facilitated by the exchange of metadata or data between software following community standards. To better convey

	this meaning, we propose updating the wording of this foundational principle: "*The software usually needs to communicate with other software via exchanged data (or possibly its metadata). Software tools can interoperate via common support for the data they exchange. "* Furthermore, we propose that the sense of direct integration is actually related to the use and reusability of software, rather than interoperability. See the discussion on Reusability foundational principle for more on this point. As part of this refocus, we will drop some guiding principles that don't reflect this, reword others, and introduce a new principle modelled on one of the reusability guiding principles.
FAIR4RS-subgroup2	*There is an opportunity to expand the second point made by Subgroup 1, that "there is an indirect and often symmetrical integration between independent software objects that can or do exchange data" with workflows.* *This is the principle of workflows and workflow management systems (WfMS) - they are expressly about the movement of data between services and the linking of inputs and outputs of codes, and those codes may be invoked on different platforms.*
FAIR4RS-subgroup4	*Interoperable software should:* ● *Be machine readable and pipeable* ● *Be able to be used together with other software and data, as part of workflows* ● *Have well-defined and documented data formats and APIs, using existing community standards where possible* ○ *This includes protocols and standards for other research objects like use of ORCID, CRediT, COPE ethical guidelines* ● *Be portable i.e. can be run (with adaptation) on similar systems, machines and environments* *Specific clarifications in response to "Towards FAIR Principles..."* ● *Unlike data, in a sense, all software is "integrated" with, or depends upon other software. And some software can be written such that it can be (easily) integrated into other software projects. Getting this right seems to be a critical component of reuse.* ● *Highly context-dependent. At best, interoperability between software and data can be discussed in the existing FAIR framework.* ● *Interoperability should also touch (together with reusable) on the property of usability. FAIR needs to stay usable - not a burden on the authors but a welcoming addition.* *Other responses on Interoperable from "Towards FAIR Principles..." suggesting additional guiding principles:*

	● *Software should document the environment required to execute the software [should this be in Reusable?]* ● *Software should support checkpointing / repetition of runs* ● *Software should be linked to related objects including publications using the code, other versions of the code, tools and libraries used, and derived versions of the code [should this be in Findable or Reusable?]*

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

Resource	Content
Applicability of principle to FAIR for Research Software	Example of use (WfMS) but level of applicability is unclear.
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 a 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."*
FAIR4RS-subgroup1	*(removed)*
FAIR4RS-subgroup2	*WfMS expressly use a language to describe the workflow - which is why they are data at one level.* *There were differing opinions on whether this was required as a guiding principle, around whether this should be enforced to encourage sharing code in broadly applicable languages that may be more usable by the community versus this being, and not coming up with languages of their own, versus this being encouraged as good practice rather than enshrining in FAIR.*
FAIR4RS-subgroup4	*Many people believed this principle applied / applied with rewriting, but some felt it did not apply to software and should be removed.* *Additional feedback, based on discussion of "Towards FAIR Principles..." rewritten version:* ● *Current phrasing of principle doesn't take into account that software is all written in a formal language, so there is some inherent standardisation, and is machine readable. Therefore this may not be relevant for software. [Compatible with Subgroup 1]* ● *For source code based software, code quality should also be considered. This is also true for the original principle with respect to knowledge representations. [Unclear if compatible with Subgroup 1]*

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

Resource	Content
Applicability of principle to FAIR for Research Software	Examples of use (CodeMeta, Citation File Format).
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 standard and the Citation File Format 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."*
FAIR4RS-subgroup1	*(removed)*
FAIR4RS-subgroup4	*1.2S.1: Many people believed this principle applied / applied with rewriting, but some felt it did not apply to software and should be removed.* *Additional feedback, based on discussion of "Towards FAIR Principles..." rewritten version:* ● *Suggested rewrite to "Software metadata are formally described using controlled vocabularies that follow the FAIR principles." [Not compatible with Subgroup 1 - principle removed]* ● *Principle applied to metadata, but unclear if it does for software. What is a controlled vocabulary for software? Is it the programming language? If yes, would any programming language be less FAIR than others? [Unclear if compatible with Subgroup 1]* ● *Controlled vocabularies are (and should be) always in progress, adapting to actual community use and practices. Software is immensely flexible and varied, it may happen that the current version of a controlled vocabulary doesn't cover a particular application that still needs documentation. So semi-formal descriptions might have a necessary role. [Unclear if compatible with Subgroup 1]* ● *Should this be moved to Reuse?* *1.2S.2: Everyone believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of "Towards FAIR Principles..." rewritten version:* ● *Difference here between type and format is unclear. [Not compatible with Subgroup 1 - principle removed]* ● *Should I2S.2 be recast as the foundational "interoperable" principle, rewritten more simply as "FAIR software should exchange*

	● *(meta)data that is FAIR"? [Unclear if compatible with Subgroup 1]* ● *Should I2S.2. be rewritten as "Software use and produce data that follows the FAIR principles."? [Not compatible with Subgroup 1 - principle removed]*

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

Resource	Content
Applicability of principle to FAIR for Research Software	Sits better in the context of reusability.
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)*
FAIR4RS-subgroup1	*I2. Software includes qualified references to other objects* This guiding principle applies to software as written, but in discussion we agreed that this is in aid of (re)use of software, rather than interoperability (at least as described above). Additionally, this simple translation of the original guiding principle doesn't capture that qualified references should be to metadata, data and software, as well as to non-digital objects that have a virtual presence in digital systems (e.g., samples, reagents, etc.). Software source code (and some other types of software) do include references to other software (requirements, imports, libraries, etc.) but not currently in a way that meets F1 and A1. Software does not generally include references to metadata, though in some cases, it can include (in comments) references to algorithms or other published text that it implements. Some software includes references to external data objects required to execute the software. To be fully FAIR, the data would ideally be FAIR as well, and references to external data fully qualified. We believe that calling for qualified references to metadata and to data is reasonable. However, in light of the modified definition of the foundational Interoperability principle, we believe that, while the inclusion of guiding principle calling for software to include qualified references to other software is reasonable, this is primarily in aid of the use and reuse of the software. For this reason, we propose that there be two guiding principles: *"Software includes qualified references to other objects"* *"Software includes qualified references to other software"*

	The second of these is a new guiding principle to be placed under the Reuseable foundational principle.
FAIR4RS-subgroup2	*This also works for workflows and scripts and all objects where the process is explicit as opposed to being buried in the code*
FAIR4RS-subgroup4	*This rewritten I2 from Subgroup 1 is compatible with the discussion around the definition of the Findable and Interoperable guiding principles from Subgroup 4.*

I4S. Software dependencies are documented and mechanisms to access them exist

Resource	Content
Towards FAIR principles for research software	*14S. Software dependencies are documented and mechanisms to access them exist.*
"5 recommendations for FAIR software"	*(not explicitly discussed)*
FAIR4RS-subgroup1	*(not explicitly discussed)*
FAIR4RS-subgroup4	*Everyone believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of "Towards FAIR Principles..." rewritten version:* ● *The important question of long-time access of dependencies is not included in the principle* ● *Dependencies describe integration, but don't automatically create the preconditions for interoperation. What (I4S) describes should be a principle for (re)use.*

New interoperability principle

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)*
FAIR4RS-subgroup1	*I1. Software should read, write or exchange data in a way that meets domain-relevant community standards* ... there is an indirect and often symmetrical integration between independent software objects that can or do exchange data. This could be in the form of information passed between two running instances of software (e.g., services), or it could be in the form of support for common data formats read or written by both software packages. This sense of integration does reflect the reciprocal

	relations expressed by interoperability.
FAIR4RS-subgroup4	*This new guiding principle from Subgroup 1 is compatible with the definition of the Interoperable foundational principle from Subgroup 4 (“Have well-defined and documented data formats and APIs, using existing community standards where possible”)*

R. Reusable

The ultimate goal of FAIR is to optimise the reuse of data. To achieve this, metadata and data should be well-described so that they can be replicated and/or combined in different settings.

Resource	Content
Towards FAIR principles for research software	*Reusability in the context of software has many dimensions. At its core, reusability aims for someone to be able to re-use software reproducibly as described by Benureau and Rougier 2018 [61]. The context of this usage can vary and should cover different scenarios: (i) reproducing the same outputs reported by the research supported by the software, (ii) (re)using the code with data other than the test one provided to obtain compatible outputs, (iii) (re)using the software for additional cases other than those stated as supported, or (iv) extending the software in order to add to its functionality.* *Software reusability depends to a high degree on software maintainability (see also Section Software quality: beyond FAIR), including proper documentation at various levels of detail. The legal framework, e.g., software licenses, is also important in terms of reusability as it determines how software can be built, modified, used, accessed and distributed. Furthermore, as research software is an integral part of the scientific process, credit attribution (citation) is another important aspect to consider with regard to (re)usability.*
"5 recommendations for FAIR software"	*Use a publicly accessible repository with version control - WHY THIS IS IMPORTANT* *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. Taken together, this ensures that your software has the best chance of being used by as many people as possible while promoting transparency.* *Add a license - WHY THIS IS IMPORTANT* *Any creative work (including software) is automatically protected by copyright. Even when the software is available via code repository 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. Finally, be aware that you, as the developer of a given piece of software, may not be a copyright owner of the code you write. Usually the copyright holder of a work is the employer (or hiring party) and not the author of the work.*
FAIR4RS-subgroup1	*The ultimate goal of FAIR is to optimize the reuse of software. To achieve this, metadata and software should be well-described so that they can be replicated and/or combined in different settings.* We believe that usability and reusability is an important foundational principle for software. However, "optimize" is too strong of a statement and should be replaced by "enable and encourage." Finally, software can be described via metadata.

	To maximise software (re)use, we must recognise that most software is dependent on other software. FAIR Research Software should be structured to maximise its potential use or reuse. This includes: ● the encapsulation of the software such that it can be reused alone or within other software projects ● the level of abstraction at which the software is expressed ● the record of references to dependencies that enable use and reuse of the software, and ● the metadata that pertains to reusability. As discussed under the interoperability foundational principle above, it has been difficult to interpret what interoperable means in a FAIR context. This is true for reusable as well. These terms have multiple, overlapping senses when applied to software. Reuse for software can mean much more than “replicated and/or combined” in the original wording for this foundational principle. We do not consider executability to be a necessary feature of software for it to be FAIR. There are many legitimate (re)uses of software that do not require executability, for instance, to verify that steps taken within the code are valid, or to look for “bugs” and other errors in the code. Software is usually written in a human readable form (source code), which will either be executed by an interpreter, or compiled into one or more binary forms suitable for execution within specific hardware and operating system combinations (limiting potential (re)use). We consider making the original human readable form available most harmonious with the FAIR principles, but recognise that for commercial, historical, or sensitivity reasons, the binary or binaries may be the only available form of some software. The binary itself is opaque and may contain bugs and errors. It is impossible to verify its validity and it cannot be modified, for example, to fix bugs. Binaries can be considered black boxes that we can “use” or “reuse” in a research workflow to produce, analyze, or act on data. Source code, on the other hand, can be interrogated, modified, and “reused” in other software or research workflows in a wider range of environments; see Gap 7 in Section 5. We suggest “replicated, combined, reinterpreted, reimplemented, and/or used” instead of “replicated and/or combined.” A new version of the text above is "The ultimate goal of FAIR is to enable and encourage the use and reuse of software. To achieve this, software should be well-described (by metadata) and appropriately structured so that it can be replicated, combined, reinterpreted, reimplemented, and/or used in different settings."

FAIR4RS-subgroup2	*The list of suggested techniques to maximise potential use or reuse are akin to the ASAP of workflows: Automation, Scaling, Abstraction, Provenance (aka dependencies). But not necessarily encapsulation. If software calls a service or an API or a microservice is it not reusable?* *If software is not required to be executable, then isn't it just data? However we agree with Subgroup 1 that reuse through reading is critical (and more sustainable than reuse through running).* *The suggested new version of the text works for workflows.*
FAIR4RS-subgroup4	*Reusable software should:* ● *Make it possible for others to understand and use the software for their own purposes* ● *Be well-documented/curated, and lower effort to use than building own* ● *Have a suitable and clear license* ● *Be usable and extensible* ● *Sustainable* ● *Reproducible* ● *Dependable* *There was considerable debate about whether the spirit of the Reusable foundational principle should concentrate on usability, enabling reuse (e.g. extensibility, maintainability, license), or reproducibility.* *An overwhelming viewpoint was that this foundational principle should encourage adherence to software engineering good practice.* *Other guiding principles suggested in this category included:* ● *Software should be written to follow software engineering principles such as encapsulation (e.g., modularity, portability, abstraction) and flexibility (e.g., less hard coded variables) to enable greater reuse* ● *Software should be written to make it easy for others to understand how to modify it* ● *Software should be written to encourage contribution (e.g. Code of conduct, contributing, readme, etc.)* ● *Software should be documented so that the intent of the software is clear (both in the code and in the documentation)* ● *Software should not contain hidden features or bugs that could compromise suitability for given tasks* ● *Software should be dependable i.e. it can be built on by other software and research*

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

Resource	Content
Applicability of principle to FAIR for Research Software	Direct examples of use
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."*
FAIR4RS-subgroup1	*R1.1. Software is richly described with a plurality of accurate and relevant attributes* This guiding principle is reasonable.
FAIR4RS-subgroup2	*Also makes sense for workflows.*
FAIR4RS-subgroup1	*Most people believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of "Towards FAIR Principles..." rewritten version:* ● *What does rich and plural mean in the context of software? [Probably incompatible with Subgroup 1]* ● *Rich metadata needs to be maintained as well, or it will be even worse than no metadata. Maybe add "up-to-date" as the first requirement and the others after that as "secondary". [Probably compatible with Subgroup 1]* ● *What do attributes mean in the context of software? More guidance is required. [Probably compatible with Subgroup 1]*

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]*
Applicability of principle to FAIR for Research Software	Direct examples of use.
"5 recommendations for FAIR software" "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."*
FAIR4RS-subgroup1	*R1.2. Software is made available with a clear and accessible software usage license* This guiding principle is reasonable, assuming that "release" is defined as making the software available. Thus, we think this principle should be re-written as "*Software is made available with a clear and accessible software usage license.*"
FAIR4RS-subgroup4	*Most people agreed with this principle as written. [Probably compatible with Subgroup 1 - needs discussion about licensing of dependencies]*

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. "*
Applicability of principle to FAIR for Research Software	Direct examples of use.
"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."*
FAIR4RS-subgroup1	*R1.3. Software is associated with detailed provenance* This guiding principle is reasonable. A version control system (VCS) may provide detailed provenance for software, but the quality of detail, especially of agents, entities and actions will depend on careful, consistent and considered use of the VCS. Also note that many contributors may not be recorded by a version control system, which by default only stores that single individual who submits each change.
FAIR4RS-subgroup2	*Also makes sense for workflows.*
FAIR4RS-subgroup4	*Everyone believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of "Towards FAIR Principles..." rewritten version:* ● *The phrase "detail level should be community agreed" just restates R1.3 [Compatible with Subgroup 1]* ● *Requires clearer definition of what "community" means [Probably compatible with Subgroup 1]* ● *Suggested rephrasing as "Software metadata include detailed provenance, detail level should be at least as high as the community agreed best practice." [Probably compatible with Subgroup 1]* ● *Provenance for software is authorship and best ensured by version control [Compatible with Subgroup 1]*

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."*
Applicability of principle to FAIR for Research Software	Consensus of applicability through careful interpretation.
"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. "*
FAIR4RS-subgroup1	*R1.3. Software meets domain-relevant community standards* This guiding principle is reasonable, but requires careful consideration for software, for the reasons in the discussion under the foundational principle and those laid out below. As noted in Section 2, one feature that differentiates software from data is that it is a complex object composed of multiple distinct objects, such as source code and/or binaries, documentation, and possibly data and metadata of various kinds (see Gaps 5 and 6 in Section 5 for more discussion). For software, the composition of the complex object may itself be subject to community standards (e.g., an expectation that certain components such as documentation or detailed references to dependencies should be included in the overall object), and the distinct objects may also be subject to separate community standards (i.e., that included or referenced objects should be in a particular form, or otherwise made FAIR in different ways). Software becomes more usable or reusable by meeting these kinds of domain-relevant community standards. Particularly when considering the source code component of software, community standards may include preferred programming languages or packaging systems. That is, the "domain-relevant community standards" include the norms established around the software community for each programming language. They also include any further norms within research domains. Community standards may include ways of managing and structuring the code, and expectations around the presence and structure of documentation; see Gap 6 in Section 5. We interpret this point as allowing multiple domains to operate at once. We do not consider it an aim of the FAIR principles for research software to pursue the integrability of all software with all software or the use of a single preferred programming language

	above all others. We also believe that, by extension, this principle can refer to the functionality or capabilities of the software, and that it is reasonable to expect that: “Software should read, write or exchange data in a way that meets domain-relevant community standards.” We note that calling for data that is read, written, or exchanged by software to be FAIR would be too strong a statement for data or metadata only used within or between a collection of software. We also do not insist that FAIR software must integrate with repository systems by default (for instance, to capture metadata and issue an identifier); we believe such decisions should be made by the software creator based on how the software will be used, in the context of community standards and expectations. This interpretation of this principle is harmonious with our proposed interpretation of Interoperability for research software. We propose that this new wording should be a new and separate principle under Interoperability (I1) in addition to preserving the original one as discussed further in Section 4.
FAIR4RS-subgroup2	*The note that “one feature that differentiates software from data is that it is a complex object composed of multiple distinct objects” is also true of some datasets. not all datasets are atomic and homogeneous. The FAIR data principles take into account that different users/stakeholders will have the need for different metadata on the same data: does that perspective apply to software?*
FAIR4RS-subgroup4	*Everyone believed this principle applied / applied with rewriting.* *Additional feedback, based on discussion of “Towards FAIR Principles...” rewritten version:* ● *R1.3, if not time bound, may be problematic. Community standards are (and should be) in constant development. Exceptions to following the standards should be possible where necessary. Suggested rephrasing to: "Software metadata and documentation meet or rise above domain-relevant community standards." [Probably compatible with Subgroup 1]* ● *The term “community standards” is fuzzy - how is this recognised? [Unclear if this is compatible with Subgroup 1 - possibly requires discussion to identify how to be documented]* ● *There should be some minimum interdisciplinary standard, as some software is not limited to a domain [Probably incompatible with Subgroup 1 as stands - does this suggest an additional principle?]*

New reusability principle - qualified references to other software

Resource	**Content**
FAIR4RS-subgroup1	*R2 Software includes qualified references to other software* Software source code (and some other types of software) do include references to other software (requirements, imports, libraries, etc.) but not currently in a way that meets F1 and A1. Software does not generally include references to metadata, though in some cases, it can include (in comments) references to algorithms or other published text that it implements. Some software includes references to external data objects required to execute the software. To be fully FAIR, the data would ideally be FAIR as well, and references to external data fully qualified.
FAIR4RS-subgroup2	*(not explicitly discussed)*
FAIR4RS-subgroup4	*This agrees with the general discussion from Subgroup 4 on the meaning of the F, A and R foundational principles.* *However Subgroup 4 goes further, and would suggest that to be fully FAIR, the software dependencies would ideally be FAIR as well.* *But, because software consists of large stacks of interdependent components, any definition of metrics and indicators of FAIR for software can only be made in the context of a specific stack. Otherwise NumPy would be criticised for not being interoperable with R.*

New reusability principle - dependability

Resource	**Content**
FAIR4RS-subgroup1	Not discussed.
FAIR4RS-subgroup2	*(not explicitly discussed)*
FAIR4RS-subgroup4	*R3. Software is dependable and can be built on by other software and research* ● *R3.1 The software is maintained by a large community, or supported by an institution that has made a long-term commitment to its maintenance.* ● *R3.2 The software comes with a policy statement about its future evolution (backward compatibility, supported platforms, etc.)* ● *R3.3 The software's dependencies are as dependable as the*

	software itself. *This should be compared with how dependability is considered for FAIR data.*

Appendix B: How to apply the FAIRsFAIR recommendations

The recommendations in the FAIRsFAIR report (Gruenpeter et al. 2020) uses the following requirement level, as defined in RFC2119[3]:

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

Recommendation number	**Recommendation**	**How to satisfy this 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.	The creation of the FAIR4RS WG is a measure to achieve this recommendation by having a dedicated discussion taking into account the FAIR guiding principles and the specific nature of software.
n°2	Applying principles and recommendations to software demands effort, time and skill. The realistic nature of these principles MUST be considered.	Having researchers who create software or research software engineers as reviewers of the principles can be a means to satisfy this 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.	Inviting specific stakeholders which were not identified in the FAIR4RS WG to review and comment on the WG outputs and the resulting FAIR principles for research software.
Recommendation n°4	Existing infrastructures that already provide solutions for software artifacts SHOULD be asked to review the FAIR principles for research software.	Invite infrastructures representatives to review the FAIR4RS principles.

3https://tools.ietf.org/html/rfc2119

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).	