FAIR-IMPACT
Expanding FAIR solutions across EOSC

This document has been submitted

Project Title	Expanding FAIR solutions across EOSC
Project Acronym	FAIR-IMPACT
Grant Agreement No.	101057344
Start Date of Project	2022-06-01
Duration of Project	36 months
Project Website	https://fair-impact.eu/

D4.4 - Guidelines for recommended metadata standard for research software within EOSC

Work Package	**WP 4, Metadata and Ontologies**
Lead Author (Org)	**Morane Gruenpeter (INRIA)**
Contributing Author(s) (Org)	**Sabrina Granger (INRIA), Alain Monteil (INRIA), Neil Chue Hong (UEDIN-SSI), Elena Breitmoser (UEDIN-SSI), Mario Antonioletti (UEDIN-SSI), Daniel Garijo (UPM), Esteban González Guardia (UPM), Alejandra Gonzalez Beltran (UKRI-STFC), Carole Goble (UNIMAN), Stian Soiland-Reyes (UNIMAN), Nick Juty (UNIMAN), Gabriela Mejias (DataCite)**
Due Date	**2023-06-30**
Date	**2023-06-30**
Version	**V1.0 DRAFT NOT YET APPROVED BY EUROPEAN COMMISSION**

Dissemination Level

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

Funded by the European Union

Versioning and contribution history

Version	Date	Author	Notes
0.1	2023.04.06	Morane Gruenpeter (INRIA)	Main structure
0.2	2023.04.21	Morane Gruenpeter (INRIA) and all co-authors	Revision of structure, metadata analysis and landscape added use cases , tools template & audience definition in introduction
0.3	2023.05.17	Morane Gruenpeter (INRIA) and all co-authors	First proposal of the RSMD recommendations
0.4	2023.05.23	Morane Gruenpeter (INRIA) and all co-authors	Draft for webinar review
0.5	2023.05.26	Morane Gruenpeter (INRIA) , all co-authors and community	Consolidated feedback
0.6	2023.05.31	Morane Gruenpeter (INRIA) and all co-authors and community	Submission to PCO and internal FAIR-IMPACT review
1.0	2023.06.30	Morane Gruenpeter (INRIA) and all co-authors and community	Submission to EC

Table of Contents

Funded by the European Union

List of Figures

List of Tables

Funded by the European Union

Terminology

Terminology/Acronym	Description
ARDC	Archival, Reference, Description and Credit
ADMS.SW	Asset Description Metadata Schema for Software
CFF	Citation File Format
CURE	Curating for Reproducibility
DOAP	Description of a Project
EC	European Commission
EOSC	European Open Science Cloud
FAIR	Findable, Accessible, Interoperable, Reusable
FAIR4RS	Findable, Accessible, Interoperable, Reusable for Research Software
FOSS	Free and Open Source Software
FSFE	Free Software Foundation Europe
IPOL	Image Processing Online
PID	Persistent Identifier
PURL	Persistent uniform resource locator
RDA	Research Data Alliance
ReSA	Research Software Alliance
RSMD	Research Software MetaData
SCID WG	Software Source Code Identification Working Group (RDA & FORCE11)
SCI WG	Software Citation Implementation Working Group (FORCE11)
SEON	Software Evolution ONtology
SIRS	Scholarly Infrastructure of Research Software
SIRS report	Refers to the Scholarly Infrastructure of Research Software report: EOSC Executive Board & EOSC Secretariat. (2020).
SPDX	Software Package Data Exchange
SSC IG	Software Source Code Interest Group
SWHID	Software Heritage Identifier
SWH	Software Heritage
T4.3	Task 4.3 in Work Package 4: *Standard metadata for research software*
VCS	Version control System
WP4	Work Package 4 in the FAIR-IMPACT project:*Metadata and ontologies*

Funded by the European Union

Executive Summary

The *FAIR-IMPACT Guidelines for recommended metadata standards for research software within the EOSC* present the first proposal *of the* Research Software MetaData (RSMD) Guidelines, developed by Task 4.3 (T4.3), *Standard metadata for research software,* as part of Work Package 4, *Metadata and ontologies,* in the FAIR-IMPACT project. FAIR-IMPACT aims to realize a FAIR (Findable, Accessible, Interoperable, and Reusable) European Open Science Cloud (EOSC) by leveraging community guidelines and existing infrastructures in the scholarly ecosystem.

The growing recognition of software's crucial role in research has led to initiatives, infrastructures and research institutions to address the challenges of software findability, accessibility, attribution and reuse. One of these initiatives, the Scholarly Infrastructures for Research Software (SIRS) Task Force (TF), identified in its report (EOSC Executive Board & EOSC Secretariat, 2020) the general need for actionable, standardized guidelines for researchers/developers that self-archive software or submit software for publication.

The FAIR-IMPACT project acknowledged the rising need for establishing software metadata guidelines to effectively collect and curate metadata. In response, T4.3's deliverable provides a comprehensive set of Research Software MetaData (RSMD) Guidelines that offer flexible and adaptable recommendations for end-users that can be used in different disciplines and different software development contexts.

This deliverable summarizes a large review and analysis including: a thorough introduction of the goals, methodology and use cases; the state of the art of existing practices and guidelines and the metadata landscape. Following the first introduction and state of the art, T4.3 introduces the RSMD guidelines proposal to collect and curate research software metadata. These guidelines are directly addressed to end users, including software creators and curators in their quest to improve the FAIRness of their software.

Lastly, T4.3 will continue its efforts to make the RSMD guidelines normative within the academic community. This includes engaging with stakeholders, gathering feedback, and incorporating best practices and advancements in metadata management within existing infrastructures. By establishing these guidelines as a norm, the aim is to promote widespread adoption and adherence, leading to greater standardization and harmonization of metadata practices across research domains. The ongoing commitment of T4.3 to refining and promoting the RSMD guidelines will contribute to the overall advancement of open and FAIR research practices in the research software community. The purpose of this deliverable in general and of the RSMD guidelines in particular is to provide a concrete and practical resource to support researchers in their endeavors to share and publish their research software creations.

Funded by the European Union

1 Introduction

Software plays a crucial role in academic research, not only as a tool for data analysis but also as a research outcome or result, or even the object of research itself (EOSC Executive Board & EOSC Secretariat, 2020). Therefore, in an Open Science ecosystem, software should receive the same level of attention and recognition as publications and datasets. However, the lack of standardized guidelines and best practices for software development and curation has resulted in challenges for researchers, developers, and other stakeholders in finding, reusing, and reproducing research results.

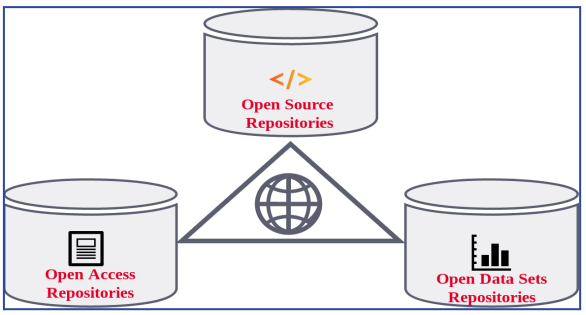


Figure 1: Three pillars of Open Science Software Heritage CC-By 4.0 2019 (Software Heritage, 2019)

"Sometimes, if you don't have the software, you don't have the data"
Christine Borgman (Turello, 2019)

As such, it is essential to ensure that software is described and shared in the same manner that researchers do with publications. The Scholarly Infrastructure for Research Software (SIRS) report (EOSC Executive Board & EOSC Secretariat, 2020) identified four major challenges that need to be addressed as ARDC:

- A for Archiving to ensure availability and accessibility
- R for Referencing to ensure precise identification for reuse and reproducibility
- D for Describing to ensure software can be found and understood
- C for Crediting to ensure proper acknowledgement of contributions and authorship

These challenges should be addressed at different levels (Nosek, 2019). At the infrastructure level, it is necessary to provide access to services that enable archival, referencing, description, and citation. Infrastructure changes require appropriate tools and user interfaces to facilitate the transition. These changes should be accompanied by new or improved community standards to establish norms. At a higher level, the proposed changes should be incentivized, and ultimately, policies can mandate their implementation.
In FAIR-IMPACT and on this specific deliverable we will address the community standard for research software metadata collection and curation.

To establish a FAIR EOSC, software should be part of this goal following the high level FAIR for Research Software (FAIR4RS) principles specified in (Chue Hong et al., 2022) where artifacts should be at least findable, accessible, interoperable, and reusable (FAIR). However "FAIR isn't the end goal, it's just one part of the solution." (D. Katz,

Funded by the European Union

2021). Other criteria should be met for software: open, archived, of high-quality, robust, citable, sustainable, and reproducible.

Task 4.3 (T4.3), *Standard metadata for research software,* is part of Work Package 4, *Metadata and ontologies,* in the FAIR-IMPACT. Its primary objective is to develop guidelines for metadata collection and curation of research software, presented in this document. The description of work available in Appendix A: T4.3 will also contribute to normalization efforts such as CodeMeta and explore synergies with initiatives like FAIR Computational Workflow principles and Bioschemas.

1.1 Scope and goals

Currently, there is a lack of readily available user-friendly tools for applying and maintaining metadata across various types of software artifacts, the *FAIR-IMPACT Guidelines for recommended metadata standards for research software within the EOSC* presented in this document aim to provide actionable, standardized guidelines that will facilitate the adoption of existing and upcoming processes and tools in the scholarly ecosystem. Task 4.3 will ensure that the guidelines developed will ease the collection and curation of metadata for research software.
Therefore, the guidelines are designed for research communities, institutions, research teams, researchers and developers who create software and self-archive software by submitting the code or/and metadata to a scholarly infrastructure.

We use the Research Software definition from the FAIR4RS output:

> Research Software includes **source code files**, **algorithms**, **scripts**, **computational workflows** and **executables** that were created **during the research process or for a research purpose**. Software components (e.g., operating systems, libraries, dependencies, packages, scripts, etc.) that are used for research but were not created during or specifically for research should be considered **software in research** and not Research Software. This differentiation may vary between disciplines. The minimal requirement for achieving computational reproducibility is that all the computational components (Research Software, software used in research, and hardware) used during the research are identified, described, and made accessible to the extent that is possible.
>
> (Gruenpeter et al., 2021)

1.1.1 Why are we here?

Ultimately, this deliverable, *FAIR-IMPACT Guidelines for recommended metadata standards for research software within the EOSC* aim to: (1) create a cohesive and standardized approach to research software metadata collection and curation that benefits all stakeholders involved; (2) help the target audience identifying key requirements and best practices that can improve the quality and reproducibility of their research software.
To do so, the approach is to compare, test and analyze the existing practices in the software development communities and in the scholarly ecosystem.

Funded by the European Union

1.1.2 Who is the intended audience of this document?

The full document, including state of the art, metadata analysis, metadata curation workflows and the FAIR-IMPACT Research Software Metadata guidelines (RSMD Guidelines) proposal, is intended for the EOSC community and metadata specialists.
The RSMD guidelines are designed to be easily adapted to national communities, specialized disciplines or software-driven communities to support efforts from research communities, research institutions and scholarly infrastructures to address their researchers or users needs.
To assist software creators in self-archiving their software and self-curating their software metadata, an RSMD checklist is included in Appendix B of this document. By using the checklist, software creators can ensure that their software is available in a suitable repository and has the necessary metadata to be easily discoverable and reusable.
The RSMD guidelines and checklist are self-contained resources that can be published independently as community-maintained documents. This approach will ensure the long-term sustainability of these resources and enhance their ability to support stakeholders in the scholarly ecosystem.

1.2 Methodology

1.2.1 Articulation of T4.3 with other initiatives

One challenge of T4.3 is seeking coordination and collaboration with various relevant projects and organizations, such as the EOSC TF "Infrastructure for Quality Research Software," the CodeMeta Initiative, the FORCE11 Software Implementation WG, the joint FORCE11, Research Data Alliance (RDA) & Research Software Alliance (ReSA) FAIR for Research Software WG (FAIR4RS WG), the RDA Software Source Code Interest Group (SSC IG), the ESFRI initiative (European Strategy Forum on Research Infrastructures, n.d.) and the FAIRCORE4EOSC EU project, which is expected to develop coordinated services and tools to archive, reference, describe and cite research software. Moreover, we will strive for alignment of the metrics developed by the task T5.2 in FAIR-impact.

In addition to these stakeholders, collaboration with publishers is critical, as it must be ensured that mainstream citation formats used by publishers, such as JATS, support all relevant software metadata items. A JATS4R recommendation for software citations (NISO JATS4R Working Group, Software Citations Subgroup, n.d.) is already available.

1.2.2 Methods used in T4.3 to create the guidelines

In order to establish practical and consistent guidelines for metadata collection and curation of research software, we adopted the following methodology:

1. **Expert subgroups** in collaboration with Task 5.2 within the FAIR-IMPACT project: These subgroups consisted of experts from different domains to review and provide feedback on the FAIR4RS principles and the ARDC (Archive, Reference, Describe, Cite) pillars from the SIRS report.

Funded by the European Union

2. Existing practices analysis with the identified seven aspects for the RSMD.
3. **Workshop** (FAIR-IMPACT, 2023): we have organized a workshop with the FAIRCORE4EOSC project to gather input and suggestions from stakeholders in the research community. The workshop focused on discussing the challenges and opportunities for metadata collection and curation of research software, and identifying the requirements and best practices that would be useful for their research software. 80 participants onsite and online have participated actively during the three hours workshop (Chue Hong, Neil, 2023; Gruenpeter, Morane, 2023).
4. **Writing sprints**: three writing sprints were organized within the T4.3 to analyze the materials and inputs from the workshop. The result of the sprints is this document.
5. **Webinar**: to review and validate the RSMD guidelines proposal, we have organized a community webinar on May 23rd 2023 which launched a community review period. The webinar was open to the research community to provide feedback and suggestions on the proposal. In Appendix I, we have captured the full list of contributors during the webinar.

The next steps to complete the open and transparent methodology:

1. **Finalization**: sharing online in an open repository the RSMD guidelines & checklist in CC-BY-4.0 to get community input and continuous improvement.
2. **Maintenance**: The document will be maintained by the RDA *SSC IG.*
3. **Dissemination**: T4.3 will disseminate the resources in the academic community through various channels, including promotion on mailing lists, presentations, and hands-on workshops.

1.3 Use cases overview

The stakeholders involved in metadata collection and curation for research software include researchers, research software engineers, and developers who serve as the creators and contributors of the software, laboratories/teams responsible for software development and maintenance, curators and librarians who play a crucial role in digital archiving and metadata management, as well as the users of the research software who rely on accurate and comprehensive metadata for discovery, evaluation, and utilization purposes.

Appendix D provides an overview of use cases for each of the above stakeholders. These use cases were collected during the Research Software workshop in March 2023 (Gruenpeter et al., 2023), with contributions from various stakeholders. The workshop highlighted the collaborative efforts of the research software community in recognizing the significance of metadata for software discovery, reproducibility, archival, and decision-making.

Funded by the European Union

2 State of The Art of Best Practices and Guidelines

Software is essential for academia, yet the majority of software is developed outside of academia in industry and developer communities (Research Data Alliance/FORCE11 Software Source Code Identification WG et al., 2020). Best practices, community standards and guidelines emerged in different domains and development communities. In this section, we will cover the main themes that are part of the prominent practices and guidelines both within and outside academia. We will divide this analysis thematically, covering:

- Software development platforms and version control systems
- Accessibility and preservation (A = Archive)
- Reference and identification (R = Reference)
- Reuse, licensing and legal aspects
- Description and classification (D = Describe)
- Credit and attribution (C = Cite)
- Re-executability: dependencies and execution environment

In the diagram below, taken from the SIRS report (EOSC Executive Board & EOSC Secretariat, 2020), the universal source code archive caters the scholarly ecosystem and its infrastructures while catering other parallel ecosystems; industry, public administration and cultural heritage.

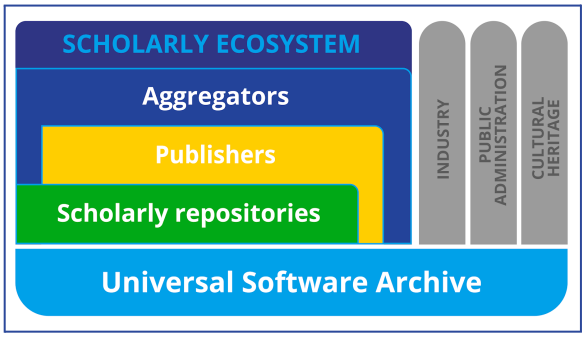


Figure 2: Architecture of interconnected scholarly infrastructures (EOSC Executive Board & EOSC Secretariat, 2020)

In this section, we will review the landscape of existing guidelines in each of the categories identified above.

2.1 Software development platforms and version control systems

The history of version control systems (VCS) dates back several decades and has evolved through different generations, transitioning from local VCS systems like Revision Control Systems to today's widely adopted distributed VCS systems like Git. Git repositories can be hosted on various platforms like GitHub, Bitbucket, and GitLab, which provide additional features like issue tracking, code review, collaboration and

Funded by the European Union

project management tools. The usage of version control systems allows to easily track changes in the software and to collaborate within a team or even in a large community. Using a VCS is strongly recommended in (Martinez-Ortiz et al., 2020) and (Chue Hong et al., 2022). Creating a code repository for the code on an online platform will generate a metadata property that should be collected: **the code repository URL**.

2.2 Accessibility and preservation

In the FAIR4RS principles (Chue Hong et al., 2022), the "A principle" stands for Accessibility: "*Software, and its metadata, is retrievable via standardized protocols.*"
Accessibility to the software cannot be guaranteed without software archival, because software artifacts are fragile. In addition, metadata should be properly archived alongside the software artifacts. It is important to note that proprietary software, which is not publicly available and cannot be publicly archived on self-archive platforms, should be archived in local instances (e.g an institutional archive).
The SIRS report (EOSC Executive Board & EOSC Secretariat, 2020) states that software archival plays a crucial role in the scholarly ecosystem and recommends connecting different scholarly infrastructures into a universal source code archive, such as Software Heritage.
GitHub or any public industrial forge is not a permanent archival solution, as many have been discontinued or closed. Scholarly repositories and the unique universal source code archive serve as separate entities dedicated to source code archival.

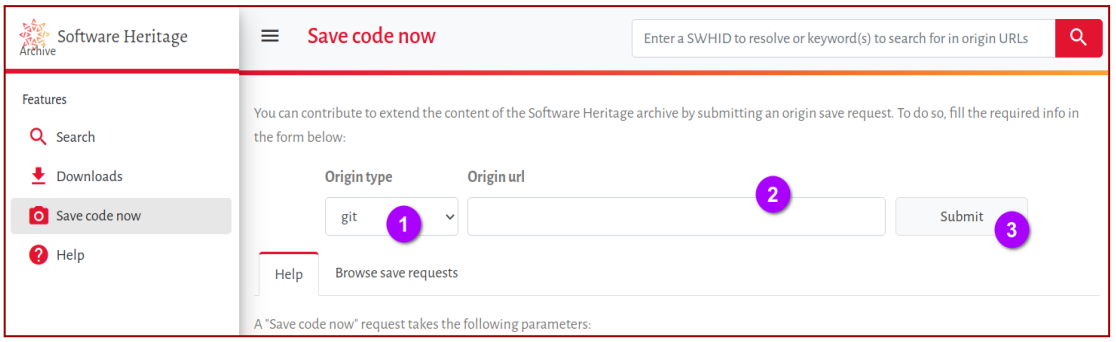


Figure 3: The "Save Code Now[1]" feature, developed by Software Heritage

In summary, the software preservation best practices are to archive the valuable code and development history which is key to its understanding and knowledge for long-term accessibility and future innovation.
By ensuring long-term access to the source code, we capture and preserve the knowledge embedded within it, serving as a valuable resource for future generations.

[1] https://www.softwareheritage.org/howto-archive-and-reference-your-code/

Funded by the European Union

2.3 Reference and identification

When it comes to software, identifying is not straightforward as one could imagine. Software is complex, it evolves over time and projects may last decades. A software project isn't a digital artifact, it is an entity that developed and produced the digital artifacts. The digital artifacts may be in source code form or executables and may be constructed from millions lines of code and may even be part of a large web of dependencies. The software projects and artifacts should be identifiable, therefore different types of Persistent Identifiers (PIDs) are required to fully meet the end-users needs (Research Data Alliance/FORCE11 Software Source Code Identification WG et al., 2020). Intrinsic identifiers, which are computed from the artifact itself are the best mechanism to reference the granularities of folder, files and code fragments.
Intrinsic identifiers, such as hashes, serve as unique signatures within the software realm. Unlike registered extrinsic identifiers, which function like passport numbers, hashes act more like fingerprints or DNA sequences for humans, providing an internal and distinct identification method.

On the other hand the intrinsic identifiers can't help identifying the projects with their metadata records, for that we need extrinsic identifiers. The analysis available in Appendix D showcases the differences between the two types of identifiers and the way to use them. In this context, using both identifiers in a citation is best, the extrinsic identifier serves to identify the metadata and attribution of the research product, while the intrinsic identifier is used to reference the precise version of the software source code associated with a deposit (Di Cosmo, Gruenpeter, Marmol, et al., 2020)

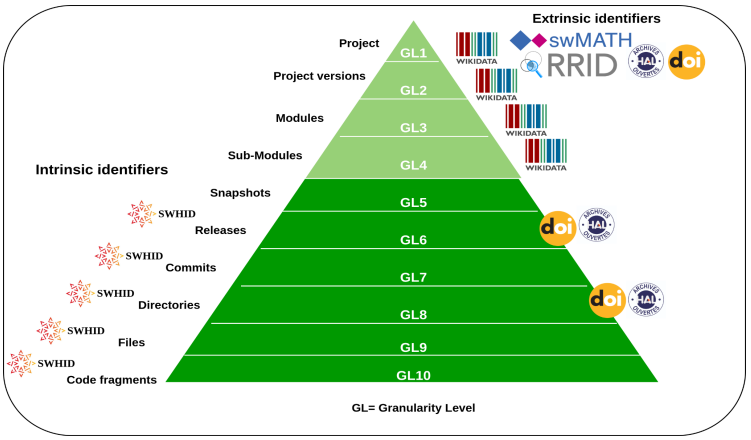


Figure 4: The granularity levels for software identifiers
from (Research Data Alliance/FORCE11 Software Source Code Identification WG et al., 2020)

In summary, implementing robust software referencing and identification practices ensures accurate and efficient retrieval of software resources for future use and collaboration.

2.4 Description and classification: to search and find software

Funded by the European Union

This section focuses on the thematic **description** for two primary use cases: (1) to understand the software better, (2) to search and find the software projects and artifacts in appropriate infrastructures. Software can only be understood and found if it contains appropriate description and metadata properties.
However, existing infrastructures utilize diverse vocabularies and ontologies to store software metadata, in Appendix G you can find the infrastructures using the CodeMeta vocabulary. A software vocabulary/ontology provides a standardized classification system for describing software artifacts, including explicit specifications of entities and relationships within a specific domain of use.

In software engineering best practices, a README file is suggested to capture general information about the software. It is also the location to note the software name and description. The README file is a human readable file and isn't machine actionable.
Best practices for providing a good README help capture essential information, facilitate understanding, and enhance the usability of the software project. For further guidance, you can refer to the resources mentioned, such as the Awesome README repository[2] and the Software Release Practice HOWTO (Raymond, 2000). It includes but isn't limited to:

1. Brief project description
2. Project website reference (if applicable)
3. Developer's build environment and potential portability issues
4. Roadmap highlighting important files and subdirectories
5. Build/installation instructions (or reference to an INSTALL file)

The descriptive information should be also captured in a machine actionable format, see metadata analysis section for intrinsic metadata files, including codemeta.json, CITATION.cff and different package managers metadata file.

2.4.1 Registries, catalogs and aggregators

Descriptive metadata about software projects include key information such as project title, description, authors/creators, contributors, keywords/subjects, and versions. The descriptive metadata can be captured in scholarly infrastructures, such as aggregators and registries, providing a concise summary of the software project, aiding in its discoverability and understanding for potential users and collaborators. By capturing and maintaining accurate and comprehensive descriptive metadata, aggregators and registries play a vital role in facilitating the visibility and accessibility of software projects within the scholarly ecosystem. Notable infrastructures in the scholarly ecosystem that handle research software are: swMATH, OpenAire, eScience center RDM, Datacite, ASCL and more. Most of these infrastructures put in place curation mechanisms to ensure metadata quality and are part of the SciCodes Consortium described in the next section.

[2] swh:1:dir:e2bff20c3bc71e81143456cad53ed599a3f2b1c3;origin=https://github.com/matiassingers/awesome-readme

Funded by the European Union

2.4.2 SciCodes Consortium and guidelines

The SciCodes Consortium is a collaborative platform for editors and maintainers of academic discipline and institutional software registries and repositories. It aims to facilitate the sharing of work methods, marketing ideas, and communication practices, as well as address challenges and find solutions in managing software resources. The Consortium focuses on promoting the adoption of standards such as CodeMeta and CFF to enhance software citation, recognition, and dissemination. Additionally, it strives to establish a virtual registry standard to enable efficient searching across multiple software registries. The consortium has developed under the FORCE 11 Software Citation Implementation WG the Best Practices for Software Registries and Repositories (Task Force on Best Practices for Software Registries et al., 2020). These guidelines are aimed at infrastructures to enhance the cataloging of scientific software, improve software discoverability, enable software citation, and promote long-term preservation and reuse of computational methods.

2.5 Credit and attribution

Software citations should aim to provide scholarly credit and normative, legal attribution to all contributors involved in the development of the software. It is important to acknowledge that different software may require varying styles or mechanisms of attribution, as a one-size-fits-all approach may not be suitable (Alliez et al., 2020). In The science code manifesto, Credit is specifically identified with the following guidance: “Software contributions must be included in systems of scientific assessment, credit, and recognition.”(The science code manifesto).
Addressing Credit is challenging even if a citation can be easily formatted, the definition of “contribution” is still very vast, as seen in (Clément-Fontaine et al., 2019)’s Recommendation n° 4 there is still the need to construct a wide and consensual definition of a "contribution" to research software. As stated in Recommendation n° 5 by Clément-Fontaine et al. (2019), it is recommended to develop tools that incorporate the concept of contribution to ensure effective crediting of authors and designers for their software contributions.
New initiatives are emerging to define a richer taxonomy for software contributions in Academia, these can be found in the software credit ontology v0.0.1[3] from 2016, in (Alliez et al., 2020) where the following software specific roles have been identified:

- Architecture
- Design
- Documentation
- Coding
- Debugging
- Maintenance
- Testing
- Support
- Management

During the Collab workshop 2023, the SORTÆD[4] Taxonomy was created for authorship definition. While the first version lacks detailed distinctions for software engineering activities, efforts in the Free and Open Source Software (FOSS) community (see the all

[3] https://w3id.org/softwareCredit#
[4] https://github.com/sdruskat/software-authorship

Funded by the European Union

contributors specifications[5]). emphasize the importance of crediting contributors. When defining a software roles taxonomy, drawing inspiration from industry engineering roles ensures alignment with established practices and enhances relevance to the software ecosystem.

2.5.1 Software citation

It is crucial to acknowledge that software has gained significant recognition, but there is still a long way to go. To elevate the status of software, institutional policies should emphasize the importance of treating software source code on par with publications and data. This can be achieved by including dedicated entries for software in bibliographies.

One important aspect of software citation is its complexity, as it requires both proper credit and identification. One recommendation is to enhance bibliographic entries by including a Software Heritage Identifier (SWHID) as an intrinsic identifier. The SWHID would resolve to the specific version of the source code archived in Software Heritage. Additionally, a persistent extrinsic identifier, such as a DOI, Hal-ID, or PURL PURL ("Persistent Uniform Resource Locator," 2023), should be included in the citation that resolves to the metadata record on a scholarly platform.

For further guidance on software citations, resources such as the "biblatex-software" package (https://ctan.org/pkg/biblatex-software) and the Software Citation Implementation Working Group (https://jats4r.org/software-citations/) can provide valuable insights and recommendations.

2.5.2 The case of the collective author

The problem arises when it is challenging to list all the authors of a software project, especially in cases where an exhaustive list is difficult or impossible to obtain. This can be the case for older software that has been taken over or when there are numerous contributors involved. In such situations, using a collective author, such as a group name, may be an option. However, it is important to ensure clarity and avoid confusion by identifying the main contributors who can be contacted for questions or further information. The recommendation in RSMD-5.4 suggests considering the inclusion of respective roles for authors to provide additional context and understanding of their contributions. See the Givaro software example (The Givaro Group, 2019), hal-02130729v1[6] as an existing example that should be improved to provide proper credit.

2.6 Reuse, licensing and legal aspect

Firstly, the software license should be as open as possible, promoting accessibility and allowing for widespread reuse. It is important to include a clear license that is both

[5] https://allcontributors.org/docs/en/specification
[6] https://hal.univ-lorraine.fr/LJK-MAD-CASC/hal-02130729v1

Funded by the European Union

human and machine-readable, ensuring easy understanding and interpretation. The use of standards like the Software Package Data Exchange (SPDX) can facilitate the licensing process and improve interoperability. These recommendations can be found in the HAL deposit guide and on Joinup[7], the platform of the European Commission's. As stated in the "The science code manifesto"("Science Code Manifesto," 2013), the copyright ownership and license of any released source code must be clearly stated." By doing so, it becomes easier for users and collaborators to understand the permissions and restrictions associated with the software.
Additionally, it is crucial to ensure that the licenses of software components used within a project are compatible with each other. This compatibility ensures that the software can be integrated and used effectively without any conflicting licensing terms.

To facilitate the long-term preservation of free software resulting from research, the creation of "legal toolboxes" is encouraged. These toolboxes may provide guidance and resources to support compliance with licenses and intellectual property laws, ensuring the continued availability and accessibility of the software. (Clément-Fontaine et al., 2019). This includes support materials such as guides, training, and assessment tools aimed at helping researchers understand the impact of their license choices and the requirement to be compliant with the dependencies they are using. By providing these resources, researchers can make informed decisions and effectively navigate legal considerations in their software development and distribution processes.

It is worth noting that even when software is available on platforms like GitHub, it is still protected by copyright, and explicit permission is required for its use. Software licenses define the rules and conditions for usage, granting permissions to users. These licenses are essential for promoting the fair and lawful use of software (Boline et al., 2021).

However, if no license is provided for a piece of software, challenges can arise in determining the rights holder, resulting in an "orphan works" problem similar to other published materials. In such cases, it may be difficult to identify and contact the rights holder (Morrissey, 2020). This issue emphasizes the importance of clear licensing and documentation to ensure the long-term preservation and proper management of software.

In summary, addressing reuse, licensing, and legal aspects of research software involves adopting open licenses, ensuring compatibility between software components, clearly stating copyright ownership and licenses, creating legal toolboxes for long-term preservation, and recognizing the challenges of identifying rights holders in some cases. By addressing these aspects, researchers can promote the reuse, accessibility, and legal compliance of their software contributions.

[7] https://joinup.ec.europa.eu/

Funded by the European Union

2.7 Re-executability: dependencies and execution environment

To ensure the ability to re-execute research software, it is important to document software dependencies in a formal, accessible, and machine-readable manner (Lamprecht et al., 2020). This involves providing precise references to the correct versions of the software source code and utilizing stable and specialized identifiers for reproducibility. By addressing these issues, researchers may facilitate the re-execution of software and improve the overall reproducibility of their work.
Links to these elements can easily break, making it difficult to maintain the stack trace. Additionally, hardware-specific software may pose challenges in terms of finding or emulating the required hardware (Gruenpeter et al. 2020). Obsolete or restricted access to operating systems can further hinder software reuse and experiment reproduction. Increasing portability through environment and hardware emulation across platforms can offer a potential solution (Software Heritage, 2023). By emulating the necessary software environments and hardware configurations, it becomes possible to run and maintain software on modern systems, ensuring continued access and usability.

The re-executability of software can be compromised when the source code becomes inaccessible or modified, especially when dependencies disappear. GNU Guix[8] came up with a solution: integrating its extensive package definitions, relying not only on source code URLs and hashes to ensure reproducibility, but also on the Software Heritage archive. SWH serves as a stable source code repository, allowing Guix to fallback and retrieve source code from the archive when the original location is unavailable.

To ensure re-executability, another aspect is proper documentation which includes the full technical requirements, including operating system, hardware, dependencies and build-instructions. The software readme.so[9], developed on github with more than 3,000 stars is providing an hosted online tool[10] to create better README including installation, running tests and more.
Even if the README is not machine actionable, it is crucial to consider the human-readable aspect of the README in guidelines, making the recommendation to provide a README file highly relevant.

[8] https://guix.gnu.org/en/blog/2019/connecting-reproducible-deployment-to-a-long-term-source-code-archive/

[9] https://archive.softwareheritage.org/swh:1:dir:d83acbe91ea4946807f6a2aa8a011780d65af151;origin=https://github.com/octokatherine/readme.so;visit=swh:1:snp:6fece9559fa5f0892c5ac63ad4f440274e7f21d9;anchor=swh:1:rev:450a6b66ecd7e645a98620dd6975b9e616156656

[10] https://readme.so/editor

Funded by the European Union

3 Metadata Analysis

In this section we explore and analyze the roles and types of metadata in software collection, curation, preservation, and usage, distinguishing between intrinsic and extrinsic metadata. We base the analysis on the diverse landscape of software metadata vocabularies and ontologies in Appendix H, including dedicated software ontologies, linked data schemes, scholarly ecosystem metadata, digital preservation schemes, and package management vocabularies. The section also introduces the CodeMeta initiative and its commitment to creating a comprehensive software vocabulary, along with tools such as the Codemeta-generator for creating and curating software metadata. *Intrinsic* and *extrinsic* metadata is defined as follows:

Intrinsic metadata	**Extrinsic metadata**
Refers to the information that is inherently embedded within a software source code artifact. It includes metadata files that are captured in the main source code directory (such as README file, codemeta.json file, etc.)	Refers to the information that is external to the software source code artifact. It includes metadata elements that provide context, provenance, and additional information about the software.

3.1 Where is the metadata available: Intrinsic vs. extrinsic metadata

Intrinsic and extrinsic metadata have distinct roles in metadata management, curation, preservation, and usage. Intrinsic metadata is often critical for ensuring the long-term preservation and usability of software, while extrinsic metadata can provide important context and provenance information that can help users find the software, understand the software, the community using it and its relation to other research outputs. For an extensive explanation of each metadata type, see Appendix E.
Intrinsic metadata are saved and indexed in the Software Heritage archive alongside the code and the full development history. Any code can be saved with the “Save code now” feature that requires no technical skills to be used. It is good practice to save all dependencies that are available even if these dependencies are not research software.

3.2 The landscape of metadata vocabularies and ontologies

The landscape of software metadata vocabularies and ontologies is diverse and includes dedicated software ontologies, linked data schemes, scholarly ecosystem metadata, digital preservation schemes, and package management vocabularies. Each approach offers a unique perspective on defining software in an engineering context and is linked to its own ecosystem (Gruenpeter et al., 2020).

The landscape is divided into two larger categories: software specific schemes, and general schemes. Vocabularies that are in both represent instances that specifically describe software in the context of a general scheme. This is the case of the *SoftwareSourceCode* class which is part of the **schema.org** schema.

Funded by the European Union

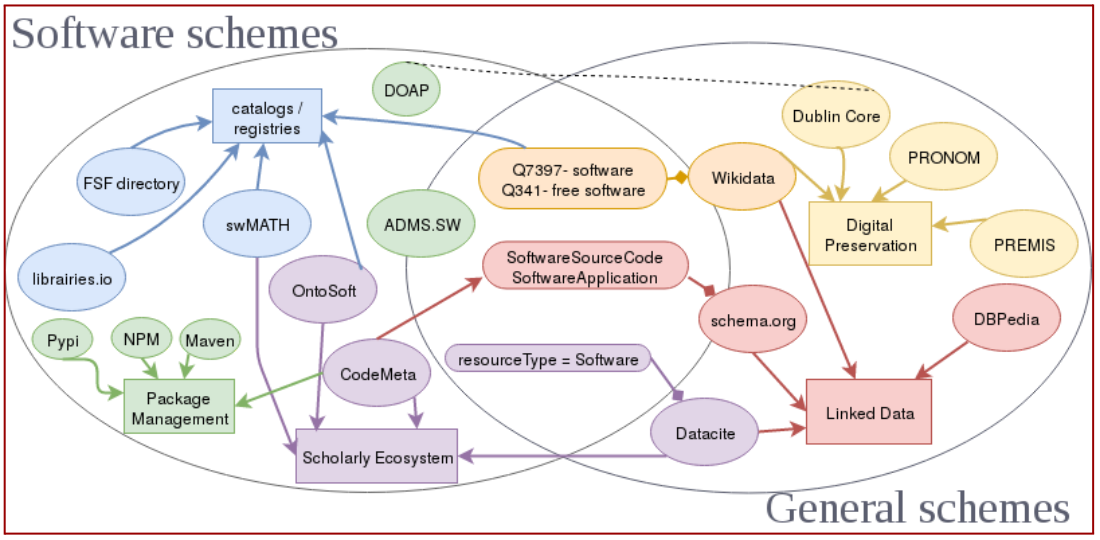
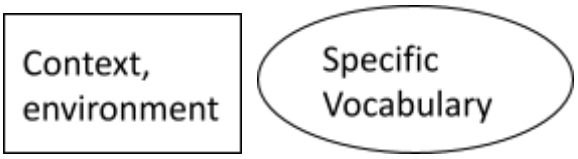


Figure 5: The metadata landscape from (Gruenpeter & Thornton, 2018)

The figure above is a non-exhaustive view of the software metadata landscape which is detailed in Appendix H. The figure's legend:



3.3 CodeMeta: Creating a Community Around a Software Vocabulary

The CodeMeta initiative was launched in 2016 with the software citation principles (Smith et al., 2016) as a result of the FORCE11 Software Citation WG. Its goal was to create a *rosetta stone* for the software vocabularies & ontologies landscape and resulted with the CodeMeta vocabulary, crosswalk table and tools. The vocabulary is built over the schema.org classes *SoftwareApplication* and *SoftwareSourceCode* which links the data for semantic web discovery.

Figure 5 presents a non exhaustive software vocabularies landscape (Gruenpeter & Thornton, 2018). In this figure, the CodeMeta vocabulary is connected to different metadata categories due to its innovative approach of providing mappings translating different software vocabularies into the CodeMeta vocabulary. These mappings, which are part of the CodeMeta repository[11]are semantic artifacts and are maintained by the CodeMeta community. In April 2023, CodeMeta v2.1[12] was released with new mappings. The CodeMeta initiative provides guidance for research software

[11] https://github.com/codemeta/codemeta
[12] Source code available:
https://archive.softwareheritage.org/swh:1:rel:03a5f4b652598aadbc59ee3cc5bf566068a44b69;origin=https://github.com/codemeta/codemeta;visit=swh:1:snp:9d85c1fdcf6fcc7d07a04022ce8ef8dea163a346
swh:1:rel:03a5f4b652598aadbc59ee3cc5bf566068a44b69;origin=https://github.com/codemeta/codemeta;visit=swh:1:snp:9d85c1fdcf6fcc7d07a04022ce8ef8dea163a346

Funded by the European Union

developers to use an intrinsic metadata file, named coedemta.json, to facilitate software metadata collection in json format.
CodeMeta is governed by the project management committee (PMC). The PMC includes eight members, in 2023. The governance model is defined by statuses which are provided via a dedicated github repository[13]. These statuses are based on a meritocratic governance model[14], by Ross Gardler and Gabriel Hanganu is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. The community is open and can contribute directly to the CodeMeta repository, by opening issues and submitting PRs. There are more than 30 contributors to CodeMeta.

As part of the WP4 activities, we will propose to the CodeMeta community the adoption of the Simple Standard for Sharing Ontology Mappings (SSSOM) in order to make the existing CodeMeta mappings[15] machine-readable and enriched with descriptive information of the mapping (e.g., who created the mapping, when, for which schema version). By adopting the SSSOM format, the mappings themselves will adhere to the FAIR principles. Furthermore, this change will allow the expression of metadata correspondence, whether two metadata properties are equivalent or just loosely related (with `*narrower than*`, `*broader than*` operators).

3.4 Tools For Metadata Creation and Curation

Tools for creating, managing, and curating software metadata play a crucial role in ensuring effective organization and discoverability of software assets. Conducting a thorough tool analysis is essential to identify and leverage the most suitable tools. In this deliverable we will describe only the codemeta-generator and call for a community effort to produce a document that will cover a large range of tools, including: codemetar[16], Bolognese[17], Codemeta-generator[18], schema.org editor[19], cffinit[20], somef[21] and cffconvert[22].

The Codemeta-generator is a web application contributed by Software Heritage during the FORCE11 2019 Research Software Hackathon. It aims to facilitate the creation of CodeMeta files (codemeta.json) by providing a user-friendly interface. The tool offers features such as form completion assistance, including a reference list of SPDX licenses, validation mechanisms, and the ability to use both CodeMeta and schema.org terms. Additionally, users can correct the output after validation as part of the creation process. The tool's purpose is to help researchers generate complete codemeta.json

[13] https://codemeta.github.io/governance/
[14] http://oss-watch.ac.uk/resources/meritocraticgovernancemodel
[15] https://archive.softwareheritage.org/swh:1:dir:26ee9cc95ffd9a3da9e1392cbd3d31946faa1fc6;
[16] https://ropensci.github.io/codemetar
[17] https://github.com/datacite/bolognese
[18] https://codemeta.github.io/create/
[19] https://schema.pythonanywhere.com/SoftwareSourceCode
[20] https://citation-file-format.github.io/cff-initializer-javascript/
[21] https://github.com/KnowledgeCaptureAndDiscovery/somef/
[22] https://pypi.org/project/cffconvert/

Funded by the European Union

files either from scratch or by aggregating existing information and adding complementary details. The target audience for this tool is researchers. Codemeta-generator is distributed under the AGPL-3.0 license and is currently in active development. You can find more information and access the tool through the GitHub repository[23] or the hosted instance on Codemeta's website[24].

4 The RSMD Guidelines Proposal for End-Users

4.1 Why do we need the RSMD guidelines?

The RSMD guidelines are a cross-disciplinary community driven effort to provide standardized and actionable guidance for the metadata collection and curation of research software. With the increasing importance of research software in academic research, it is necessary to ensure that software is archived, referenced, described and cited (ARDC) and that its metadata is findable, accessible, interoperable, and reusable (FAIR). The RSMD guidelines provide a framework for achieving these goals and facilitating the quality and sustainability of research software. Additionally, the guidelines can help stakeholders in the research community, such as researchers, software developers, research institutions, publishers, and infrastructure managers, identify requirements and best practices that are useful for their software.

4.2 Who is it for?

The RSMD guidelines are designed to be easily adapted to national communities, specialized disciplines or software driven communities to support efforts from research communities, research institutions and scholarly infrastructures to address their researchers or users needs. To assist **software creators** in self-archiving and self-metadata curation, an RSMD checklist is included in the Appendix B[25] of this document. By using the checklist, software creators can ensure that their software has the necessary metadata to be easily discoverable and reusable.

4.3 What is the scope?

The RSMD guidelines provide a standardized community based requirements and recommendations to end users in the following areas:

- General metadata requirements
- Accessibility and preservation (A = Archive)
- Reference and identification (R = Reference)
- Description and classification (D = Describe)
- Credit and attribution (C = Cite)
- Reuse, licensing and legal aspects
- Re-executability: dependencies and execution environment

[23] https://github.com/codemeta/codemeta-generator
[24] https://codemeta.github.io/codemeta-generator/)
[25] https://docs.google.com/document/d/1A4toAOJI5CJx-S2KrA6-BpWavaPS_MMayuWPJi73l78/edit#heading=h.25aaoqkhwc6a

Funded by the European Union

While the guidelines cover a broad range of aspects, they primarily focus on metadata collection, management, and curation, and do not address software engineering practices such as software quality, testing, performance optimization, or collaboration.

The following elements are considered out of scope for this document:
1. Skills: While important, the discussion of end-users skills and training falls beyond the scope of this document.
2. Homogeneity between disciplines and user profiles: Addressing the heterogeneity of disciplines and user profiles is beyond the scope of this document.
3. Documentation: Although essential for comprehensive software, the specific guidance and best practices on documenting code fall outside the perimeter of this document.

4.4 The recommendations

The RSMD guidelines are organized into seven distinct aspects, each with a clear high-level objective and a set of actionable and detailed recommendations. Each recommendation is uniquely identified using the semantic identifier `RSMD-X.Y`, where X represents the aspect number and Y represents the recommendation number. Priorities are assigned to each recommendation, categorized into three levels: essential (highest priority), important, and useful (lowest priority). This prioritization helps to emphasize the critical recommendations and ensure that the guidelines address key areas effectively.

Funded by
the European Union

4.4.1 General Metadata Recommendations

Objective		
To ensure the collection, curation, and maintenance of research software metadata, the following general recommendations are suggested for end users, including researchers, software engineers, curators, and institution staff.		
ID	**Recommendation**	**Priority**
RSMD-1.1	Provide embedded information about software in the code itself as a metadata file (intrinsic metadata) to ensure it is kept and maintained alongside the software. Can be applied to open and proprietary software. Definition: • **Intrinsic metadata** refers to information that is embedded within the content or object itself, in a metadata file that can be human readable (e.g README) or machine-actionable (e.g pom.xml, codemeta.json)	Essential ☆☆☆
RSMD-1.2	Issue a metadata record (extrinsic metadata) about the source code in a publicly available infrastructure to benefit from the infrastructure metadata service (such as citation exports and search functionalities): • scholarly repository (e.g HAL, Zenodo, etc.) • publisher (e.g IPOL, eLife, Dagstuhl[26], etc.) • registry/ aggregator (e.g ASCL, swMath, WikiData, DataCite, bio.tools, etc.) Definition: • **Extrinsic metadata** is the record describing the software externally to the source code.	Essential ☆☆☆
RSMD-1.3	Use a version control system to track changes in the source code development and facilitate archival. If an online platform is used, capture the URL in a **CodeRepository** property.	Important ☆☆
RSMD-1.4	Follow community standards when developing software: An example can be language-specific standards and best practices (e.g. DESCRIPTION file in R, see also https://swcarpentry.github.io/r-novice-inflammation/08-making-packages-R.html).	Useful ☆
RSMD-1.5	Keep the machine readable intrinsic metadata information in a single source file, and automatically generate other file formats, to avoid misalignment and duplication (e.g. generate citation.cff from codemeta.json) wherever possible.	Useful ☆

[26] https://www.dagstuhl.de/

Funded by the European Union

4.4.2 Accessibility and preservation

Objective
To ensure accessibility and preservation, researchers and software engineers are strongly recommended to follow the archival and sharing recommendations below.

ID	Recommendation	Priority
RSMD-2.1	Archive the source code repository in the universal source code archive, Software Heritage[27], using the save code now,[28] to ensure long-term access to the full development history. • This method can be used for software at any level of maturity when software is publicly available.	Essential ☆☆☆
RSMD-2.2	Archive software source code in a scholarly repository (e.g Zenodo[29], HAL[30]) to ensure that software can be found and accessed in a *scholarly context*. Note: Proprietary software can be deposited in an institutional repository. • This method should be used when a software object is ready to be reported and shared. Some repositories offer the possibility to create a software record with the archived copy in RSMD-2.1 (e.g HAL[31]).	Essential ☆☆☆
RSMD-2.3	Register software projects in a disciplinary or community registry (e.g ascl.net[32], bio.tools[33], swMath[34], RRID portal[35], RSD[36], WikiData[37], DataCite[38]) to ensure that software can be found and accessed. To answer this recommendation, semi-automatic tools should be provided by infrastructures to reduce duplication of effort done by researchers or software creators. Note: Proprietary software can be registered in an institutional catalog (e.g the INRIA BIL).	Important ☆☆

[27] https://archive.softwareheritage.org/
[28] save.softwareheritage.org
[29] https://zenodo.org/
[30] https://hal.science/
[31] https://www.softwareheritage.org/2023/04/04/swhid-deposit-hal/
[32] https://ascl.net/
[33] https://bio.tools/
[34] https://zbmath.org/software/
[35] https://scicrunch.org/resources/about/resource
[36] https://research-software-directory.org/
[37] https://www.wikidata.org/wiki/Wikidata:Main_Page
[38] https://datacite.org/

Funded by the European Union

4.4.3 Reference and identification

Objective		
To ensure that research software projects, modules, versions and source code artifacts can be precisely identified and referenced.		
ID	**Recommendation**	**Priority**
RSMD-3.1	Versioning – provide information about software versions to ensure clear identification when referencing or citing software. The usage of a version control system facilitates tracking versions.	Essential ☆☆☆
RSMD-3.2	Intrinsic identifiers – use existing well-established intrinsic identifiers (e.g SWHID[39]) to specifically identify software artifacts (e.g code-fragment, file, directory, revision, release) and dependencies.	Essential ☆☆☆
RSMD-3.3	Extrinsic identifiers – use existing well-established extrinsic identifiers (e.g DOI, HAL-ID, Wikidata entity) to identify *software version*, *software module* or *software project*. Definitions: ● **software version**: A specific version of a software refers to a distinct release or iteration of a software application or program. ● **software module**: A specific module of a larger software project refers to a self-contained unit or component within a software system. ● **software project**: A computer software project is an organized effort to develop software.	Essential ☆☆☆
RSMD-3.4	If applicable, use a versioning scheme that is accepted in the relevant community (semantic versioning, calendar versioning, git tags, etc.).	Useful ☆
RSMD-3.5	Granularity level – when identifying software artifacts use appropriate granularity level, as shown in figure 4, depending on the use case (see Appendix C: Metadata use case collection[40]).	Important ☆☆

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

[40] https://docs.google.com/document/d/1A4toAOJI5CJx-S2KrA6-BpWavaPS_MMayuWPJi73l78/edit#heading=h.6qo8c2t40npo

4.4.4 Description & classification

Objective
To ensure software findability and comprehensibility, provide descriptive metadata (software's name, purpose, functionalities, programming language, domain, etc.). These metadata facilitate accurate representation of the software and enable users to easily discover and understand its capabilities.

ID	Recommendation	Priority
RSMD-4.1	Add **software name** and **description** of the software's functionality and purpose, using a README file in the root directory of the source code or other intrinsic metadata file (e.g codemeta.json with the properties *name* and *description*) and on the metadata record (extrinsic metadata). ● Name ● Description Note: Divergences should be avoided as much as possible.	Essential ☆☆☆
RSMD-4.2	Add **descriptive metadata** for classification purposes embedded in the code (intrinsic metadata) or/and on metadata record (extrinsic metadata). This includes, but is not limited to: domain, programming language, date created, date of first publication, keywords, related links, etc. Note: There are existing infrastructures that can read a codemeta.json file or other metadata file as input to create the software record automatically (see Appendix G).	Important ☆☆
RSMD-4.3	Cite related resources (e.g journal articles) describing the software with a persistent identifier or stable URL in README file or in other intrinsic metadata files (e.g codemeta.json with the property *referencePublication*).	Important ☆☆
RSMD-4.4	Add descriptive metadata of the software in a machine readable format, for example codemeta.json or a package manager metadata file. To avoid duplication of information refer to RSMD-1.5.	Important ☆☆
RSMD-4.5	Add descriptive information in README file or other intrinsic metadata file as proposed in community standards (e.g Software Release Practice HOWTO and Make a README) ○ Website ○ Link to the documentation ○ Contact & support ○ List of functionalities ○ Development status, e.g. Active, inactive, suspended. See repostatus.org	Useful ☆

Funded by the European Union

4.4.5 Credit & attribution

Objective
To ensure proper crediting and acknowledgment of software creators, authors, and contributors, it is important to follow citation recommendations.

ID	Recommendation	Priority
RSMD-5.1	Add intrinsic metadata about authors, this information can be captured in a human readable format(AUTHORS file, as a section in README, CONTRIBUTORS.md etc.) or in an author's property in a machine-readable format (e.g codemeta.json, package management file, etc.).	Essential ☆☆☆
RSMD-5.2	Add authors and contributors to the metadata record.	Essential ☆☆☆
RSMD-5.3	Use people identifiers for non-ambiguous attribution (ORCID, ID-HAL, ID-REF, GPG key, VIAF, ISNI etc.)	Important ☆☆
RSMD-5.4	Use roles for software actors (authors & contributors). Refer to existing contributor roles if possible: - https://allcontributors.org/ - CRediT – Contributor Roles Taxonomy (niso.org) - SORTÆD: Software Role Taxonomy and Authorship Definition - https://w3id.org/softwareCredit#	Important ☆☆
RSMD-5.5	Provide citation preference (CITATION.cff file, Citation section in README, .bib file, etc.)	Important ☆☆
RSMD-5.6	Where applicable, use appropriate citation granularity level (software, software-module, software-version, code-fragment) by referring to the BibLaTeX[41] software package specifications.	Useful ☆
RSMD-5.7	In case of plurality of authors, identify all the owners : the authors list should be exhaustive to ensure credit. Note: in some cases it might be impossible to have an exhaustive list of authors, in this case a collective author may be used.	Useful ☆

[41] https://ctan.org/pkg/biblatex-software ;
https://fr.overleaf.com/learn/latex/Bibliography_management_with_biblatex

Funded by the European Union

4.4.6 Reuse, licensing and legal aspects

Objective
To ensure proper software reuse and license compliance, it is essential to accurately describe software licensing and legal aspects. This includes providing clear guidance on proper usage and distribution rights, clarifying the terms and conditions under which the software can be used and shared.

ID	Recommendation	Priority
RSMD-6.1	Authors of the software SHOULD/MUST determine the right holders and check their institution's policy before choosing a license. It is important to note that researchers are bound to their institution's policy regarding software licensing.	Essential ☆☆☆
RSMD-6.2	Add a License to the software source code as intrinsic metadata, following established best practices (e.g. SPDX license identifiers in file: headers, copy of the chosen license in a LICENSE* or in a LICENSES folder, see REUSE specification). file, README.	Essential ☆☆☆
RSMD-6.3	Add extrinsic metadata about the software license, copyright holders, and embargo end date on the metadata record.	Essential ☆☆☆
RSMD-6.4	Identify external software modules used, their authors (right holders) and their licenses. For compliance purposes, it is necessary to verify licenses compatibility.	Important ☆☆
RSMD-6.5	Add file headers the name of the software, year, copyright holder, contact information (email), license identifier (e.g. SPDX[42]), authors (and information related to modification of authors ; e.g. author BB – Date WW, Author DD – Date UU) as specified in REUSE specifications[43]:	Useful ☆

[42] https://spdx.org/licenses/
[43] https://reuse.software/spec/

Funded by the European Union

4.4.7 Re-execute: dependencies and execution environment

Objective		
To ensure the usability of software and the ability to reproduce the same results in experiments, it is important that the software can be easily rebuilt and executed. This ensures that others can use the software effectively and achieve consistent outcomes.		
ID	**Recommendation**	**Priority**
RSMD-7.1	Dependencies – describe software dependencies. If possible, provide a machine readable description (e.g. for a package manager or build system used in the specific ecosystem).	Essential ☆☆☆
RSMD-7.2	Operating system – describe the technical requirements of the environment to use the software following community practices of the specific ecosystem.	Essential ☆☆☆
RSMD-7.3	Hardware – describe technical requirements of the hardware platform, following community practices of the specific ecosystem.	Important ☆☆
RSMD-7.4	Build instructions – provide build instructions to ensure compilation and execution. Whenever possible, provide the appropriate build configuration files for the specific ecosystem (Makefile, Ant files, Dockerfile, etc.), and appropriate test cases. Following the ecosystem community standards.	Important ☆☆
RSMD-7.5	User documentation – Provide documentation for the user, including e.g. input/output formats, examples etc. Add a link to the documentation in descriptive metadata.	Important ☆☆
RSMD-7.6	Data – provide information on how the data used and/or produced by the software is handled. If applicable or available, add a sample provenance trace/log of an execution, including pointers to input data and expected results, so that potential reusers can compare and check if the obtained results are correct. This particularly applies to computational workflows & pipelines, an important subclass of research software.	Useful ☆

Funded by the European Union

5 Making it Possible: Scholarly Infrastructures' Role

While analyzing the state of the art and creating the RSMD guidelines, we encountered actions, recommendations and requirements that are not in scope for the RSMD guidelines but are needed to make metadata curation a reality.
Scholarly infrastructures, listed in Appendix F, should provide platforms that support the community practices and the RSMD guidelines.

> *"Policy describes what is required, desired, and incentivised;*
> *infrastructure determines what is possible;*
> *but the community determines how things are done in practice."*
> (Brinkman et al., 2020)

In the list below we have collected actions and requirements that should be executed by scholarly infrastructures (e.g scholarly repositories, aggregators and publishers) using the RSMD thematics, note that this list isn't exhaustive:

- ➔ **General requirements**:
  - Ensure metadata records are licensed under CC0
  - Provide mechanisms to moderate and curate metadata to ensure curation quality of software records in the infrastructure collection
  - Provide specific guidance on how to use the infrastructure services to ensure metadata is collected and preserved
- ➔ **Accessibility & preservation**:
  - Follow scholarly best practices identified by the scicodes consortium[44] to ensure quality service, this includes but is not limited to: Providing a public scope statement, disclosing infrastructure end-of-life policy and stipulating conditions of use
- ➔ **Reference and identification**:
  - Provide tools to track the software in other scholarly outputs: aggregating information about entities and the relationships between them enables strategic analysis (Fenner, 2020)
  - Publish policies for research products including PIDs for software
  - Provide guidance to article authors to identify which PID should be used to identify software projects or artifacts: extrinsic and intrinsic identifiers
  - Implement platforms and services with better and consistent PID integrations and help grow the PID Graph to track connections.
  - Cross-link a plurality of academic outputs at the infrastructure level by providing adapted metadata properties

[44] https://scicodes.net/

Funded by the European Union

➔ **Description & classification**

- Share the metadata schema, taxonomy, vocabulary or ontology used by the infrastructure to store the metadata
- Provide import and export metadata features using the codemeta.json format

➔ **Attribution & credit**

- Provide citation export formats to facilitate credit (e.g BibTeX export format)
- Establish clear authorship policy acceptable by the infrastructure (most relevant to publishers)
- Provide mechanisms to cite collective author

➔ **Reuse, licensing and legal aspects**:

- Provide standardized method for display copyright and licensing information between projects and people; e.g.: SPDX licenses list
- Require license information using a controlled list (SPDX licenses list)
- Support software authors by providing guidance on license use, etc.

➔ **Re-execute: dependencies and execution environment**

- Provide metadata properties to store information about operating system, hardware and dependencies
- Provide guidance, if possible, on how to ensure better sustainability of dull software stack (e.g GUIX)

In Appendix G, we have outlined the progress of CodeMeta adoption within prominent scholarly repositories and outlined their future plans to achieve CodeMeta compatibility.

6 Conclusion and Next Steps

With this deliverable, T4.3 presents the initial proposal of the FAIR-IMPACT Research Software MetaData (RSMD) Guidelines to the EOSC community and the wider scholarly ecosystem. These guidelines focus on end-users, including researchers, software creators, and curators, during the process of collecting, curating, and managing software metadata. However, they are also applicable to various stakeholders, such as infrastructures that offer the required platforms, services, and tools to adhere to these guidelines.

The RSMD guidelines were first introduced in the FAIR-IMPACT webinar, on May 23rd 2023. The community comments following the webinar were integrated in this version. The RSMD guidelines and checklist will be made available as standalone resources on

Funded by the European Union

an open and collaborative platform with CC-BY-4.0 license. These resources will be maintained by the community, through the RDA *Software Source Code Interest Group* (SSC IG) to ensure sustainability and foster widespread adoption across research domains.

Recognizing the significance of research software, these guidelines provide comprehensive recommendations for the collection and curation of metadata. They address challenges related to archiving, referencing, describing, and citing software, as well as guidance for ensuring the findability, accessibility, interoperability, and reusability of software within the scholarly ecosystem. The ultimate goal of these guidelines is to establish a FAIR EOSC that treats software on an equal footing with publications and data.

References

APA 7th ed. ; https://www.zotero.org/groups/5018631/fair-impact_t4.3/library

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

American Geophysical Union. (n.d.). Data and Software for Authors. *American Geophysical Union*. Retrieved April 6, 2023, from https://www.agu.org/Publish-with-AGU/Publish/Author-Resources/Data-and-Software-for-Authors

Artaza, H., Chue Hong, N. P., Corpas, M., Corpuz, A., Hooft, R. W. W., Jiménez, R. C., Leskošek, B., Olivier, B. G., Stourac, J., Vařeková, R. S., Parys, T. V., & Vaughan, D. (2016). *Top 10 metrics for life science software good practices* (5:2000). F1000Research. https://doi.org/10.12688/f1000research.9206.1

Association for Computing Machinery. (2018). *Artifact Review and Badging*. https://www.acm.org/publications/policies/artifact-review-badging

Boline, J., Das, S., Faye, I., Goscinski, W., Hellgren-Kotaleski, J., Hicks, R., Kennedy, D., Leergaard, T., Martone, M., Mouček, R., Venugopal, S., Wachtler, T., & Abrams, M. (2021). Best practices review of five recommendations for FAIR software. *F1000Research*, *10*(195), Article 195. https://doi.org/10.7490/f1000research.1118518.1

Bouquin, D. R., Chivvis, D. A., Henneken, E., Lockhart, K., Muench, A., & Koch, J. (2020). Credit Lost: Two Decades of Software Citation in Astronomy. *The Astrophysical Journal Supplement Series*, *249*(1), 8. https://doi.org/10.3847/1538-4365/ab7be6

Brack, P., Crowther, P., Soiland-Reyes, S., Owen, S., Lowe, D., Williams, A. R., Groom, Q., Dillen, M., Coppens, F., Grüning, B., Eguinoa, I., Ewels, P., & Goble, C. (2022). Ten simple rules for making a software tool workflow-ready. *PLOS Computational Biology*, *18*(3), e1009823. https://doi.org/10.1371/journal.pcbi.1009823

Brinkman, L., Schettino, A., Zurita-Milla, R., Eerland, A., Imming, M., Zormpa, E., Schwamm, H., Heininga, V. E., Tsang, E., & van 't Veer, A. (2020). *Open Science Community Starter Kit* (A. Blanchard, Trans.). International Network of Open Science & Scholarship Communities (INOSC). https://osf.io/7vez3

Canteaut, A., Fernández, M. A., Maranget, L., Perin, S., Ricchiuto, M., Serrano, M., & Thomé, E. (2021). *Software Evaluation* [Report]. Inria. https://hal.inria.fr/hal-03110728

Chue Hong, N. P., Allen, A., Gonzalez-Beltran, A., de Waard, A., Smith, A. M., Robinson, C., Jones, C., Bouquin, D., Katz, D. S., Kennedy, D., Ryder, G., Hausman, J., Hwang, L., Jones, M. B., Harrison, M., Crosas, M., Wu, M., Löwe, P., Haines, R., … Pollard, T. (2019). *Software Citation Checklist for*

Funded by the European Union

Authors (0.9.0). FORCE11 Software Citation Implementation Working Group. https://doi.org/10.5281/zenodo.3479199

Chue Hong, N. P., Katz, D. S., Barker, M., Lamprecht, A.-L., Martinez, C., Psomopoulos, F. E., Harrow, J., Castro, L. J., Gruenpeter, M., Martinez, P. A., Honeyman, T., Struck, A., Lee, A., Loewe, A., van Werkhoven, B., Jones, C., Garijo, D., Plomp, E., Genova, F., … WG, R. F. (2022). *FAIR Principles for Research Software (FAIR4RS Principles) (1.0)*. https://doi.org/10.15497/RDA00068

Chue Hong, Neil. (2023, March 24). *An overview of FAIR4RS and existing tools to assess FAIRness of software*. Research Data Alliance Plenary Session, Gothenburg. Zenodo.https://doi.org/10.5281/zenodo.7805608

Clément-Fontaine, M., Di Cosmo, R., Guerry, B., Moreau, P., & Pellegrini, F. (2019). *Encouraging a wider usage of software derived from research. Note the from Committee for Open Science's Free Software and Open Source Project Group*. French Committee for Open Science. https://hal-lara.archives-ouvertes.fr/OUVRIR-LA-SCIENCE/hal-02545142

CodeRefinery. (n.d.). CodeRefinery: Training and e-Infrastructure for Research Software Development. *CodeRefinery*. Retrieved April 6, 2023, from https://coderefinery.org/

Crusoe, M. R., Abeln, S., Iosup, A., Amstutz, P., Chilton, J., Tijanić, N., Ménager, H., Soiland-Reyes, S., Gavrilović, B., Goble, C., & Community, T. C. (2022). Methods included: Standardizing computational reuse and portability with the Common Workflow Language. *Communications of the ACM*, *65*(6), 54–63. https://doi.org/10.1145/3486897

Crusoe, M. R., Abeln, S., Iosup, A., Amstutz, P., Chilton, J., Tijanić, N., Ménager, H., Soiland-Reyes, S., Gavrilović, B., Goble, C., & The CWL Community. (2022). *Common Workflow Language User Guide*. https://www.commonwl.org/user_guide/

CURE Consortium. (2021, 2022). Curating research artifacts to support scientific integrity. *CUrating for REproducibility*. https://curating4reproducibility.org/

D4.4_FAIR-IMPACT_Guidelines_research_software_2023-04-21_v0.3. (n.d.). Google Docs. Retrieved May 17, 2023, from https://docs.google.com/document/d/1A4toAOJI5CJx-S2KrA6-BpWavaPS_MMayuWPJi73l78/edit?usp=share_link&usp=embed_facebook

Davidson, J., Grootveld, M., Verburg, M., van Horik, R., O'Connor, R., Engelhardt, C., Garbuglia, F., Vieira, A., Newbold, E., Proudman, V., & Horton, L. (2022). *FAIR-enabling Data Policy Checklist*. Zenodo. https://doi.org/10.5281/zenodo.6225775

Di Cosmo, R. (2019). *How to use Software Heritage for archiving and referencing your source code: Guidelines and walkthrough*. https://hal.archives-ouvertes.fr/hal-02263344

Di Cosmo, R., & Gruenpeter, M. (2020, January 29). *The SWH-ID: A digital fingerprint identifying software source code*. Zenodo.https://doi.org/10.5281/zenodo.3630124

Di Cosmo, R., Gruenpeter, M., Marmol, B., Monteil, A., Romary, L., & Sadowska, J. (2020). Curated Archiving of Research Software Artifacts: Lessons Learned from the French Open Archive (HAL). *International Journal of Digital Curation*, *15*(1), 16. https://doi.org/10.2218/ijdc.v15i1.698

Di Cosmo, R., Gruenpeter, M., & Zacchiroli, S. (2020). Referencing Source Code Artifacts: A Separate Concern in Software Citation. *Computing in Science & Engineering*, *22*(2), 33–43. https://doi.org/10.1109/MCSE.2019.2963148

Druskat, S. (n.d.). *SORTÆD: Software Role Taxonomy and Authorship Definition*. Retrieved May 10, 2023, from https://sdruskat.net/software-authorship/#roles

Earth Science Information Partners (ESIP) Software and Services Citation Cluster, Hausman, J., Stall, S., Gallagher, J., & Wu, M. (2019). *Software and Services Citation Guidelines and Examples*. ESIP. https://doi.org/10.6084/m9.figshare.7640426.v4

EarthCube OntoSoft project. (n.d.). *The Geoscience Paper of the Future Initiative*. Retrieved April 6, 2023, from https://www.scientificpaperofthefuture.org/gpf/

EOSC Executive Board & EOSC Secretariat. (2020). *Scholarly infrastructures for research software. Report from the EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS*. European Commission. Directorate General for Research and Innovation. https://data.europa.eu/doi/10.2777/28598

European Strategy Forum on Research Infrastructures. (n.d.). *RI Initiatives*. European Strategy Forum on Research Infrastructures. Retrieved May 31, 2023, from https://www.esfri.eu/ri-initiatives

Funded by the European Union

FAIR-IMPACT. (2023, May 23). *Developing Guidelines for Metadata Collection and Curation for Research Software*. FAIR-IMPACT. https://fair-impact.eu/events/fairimpact-events/developing-guidelines-metadata-collection-and-curation-research-software

Fenner, M. (2020, August 27). DataCite Commons – Exploiting the Power of PIDs and the PID Graph. *DataCite Blog*. https://blog.datacite.org/power-of-pids/

Garijo, D., Ménager, H., Hwang, L., Trisovic, A., Hucka, M., Morrell, T., & Allen, A. (2022). Nine best practices for research software registries and repositories. *PeerJ Computer Science*, *8*, e1023. https://doi.org/10.7717/peerj-cs.1023

Gil, Y., Ratnakar, V., & Garijo, D. (2015). OntoSoft: Capturing Scientific Software Metadata. *Proceedings of the 8th International Conference on Knowledge Capture*, 1–4. https://doi.org/10.1145/2815833.2816955

Goble, C., Cohen-Boulakia, S., Soiland-Reyes, S., Garijo, D., Gil, Y., Crusoe, M. R., Peters, K., & Schober, D. (2020). FAIR Computational Workflows. *Data Intelligence*, *2*(1–2), 108–121. https://doi.org/10.1162/dint_a_00033

Gruenpeter, M. (2023, May 23). *Developing Guidelines for Metadata Collection and Curation for Research Software*. FAIR-IMPACT webinar, Online. https://doi.org/10.5281/zenodo.7962734

Gruenpeter, M., Chue Hong, N., Granger, S., Breitmoser, E., Priddy, M., Antonioletti, M., Martinez, P. A., Honeyman, T., Collins, J. A., & Meneses, R. (2023). *Research Software Workshop: Guidelines and Metrics for Metadata Curation*. https://doi.org/10.5281/zenodo.8075097

Gruenpeter, M., Sadowska, J., Nivault, E., & Monteil, A. (2022). *Create software deposit in HAL* [Technical Report]. Inria ; CCSD ; Software Heritage. https://hal.inria.fr/hal-01872189

Gruenpeter, M., & Thornton, K. (2018, March 25). *Pathways for Discovery of Free Softwares*. Wikidata for Digital Preservation, Cambridge. https://en.wikipedia.org/wiki/File:Pathways-discovery-free.pdf

Gruenpeter, Morane. (2023, March 24). *An overview of the metadata landscape & descriptive metadata curation*. Research Data Alliance Plenary Session, Gothenburg. Zenodo.https://doi.org/10.5281/zenodo.7771642

Gruenpeter, Morane, Katz, Daniel S., Lamprecht, Anna-Lena, Honeyman, Tom, Garijo, Daniel, Struck, Alexander, Niehues, Anna, Martinez, Paula Andrea, Castro, Leyla Jael, Rabemanantsoa, Tovo, Chue Hong, Neil P., Martinez-Ortiz, Carlos, Sesink, Laurents, Liffers, Matthias, Fouilloux, Anne Claire, Erdmann, Chris, Peroni, Silvio, Martinez Lavanchy, Paula, Todorov, Ilian, & Sinha, Manodeep. (2021). *Defining Research Software: A controversial discussion* (Version 1). Zenodo. https://doi.org/10.5281/ZENODO.5504016

Hendricks, G. (n.d.). *Data and software citation deposit guide* [Website]. Crossref. Retrieved April 6, 2023, from https://www.crossref.org/documentation/reference-linking/data-and-software-citation-deposit-guide/

International Neuroinformatics Coordinating Facility (INCF). (n.d.). *Evaluation Criteria Checklist for Repositories and Scientific gateways*. International Neuroinformatics Coordinating Facility (INCF). Retrieved November 28, 2022, from https://www.incf.org/criteria-checklist

Ison, J., Kalaš, M., Jonassen, I., Bolser, D., Uludag, M., McWilliam, H., Malone, J., Lopez, R., Pettifer, S., & Rice, P. (2013). EDAM: An ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, *29*(10), 1325–1332. https://doi.org/10.1093/bioinformatics/btt113

Jisc. (2018, March 30). *Research data management toolkit*. Research Data Management Toolkit. https://www.jisc.ac.uk/guides/rdm-toolkit

Katz, D. (2021, June 28). FAIR is not the end goal. *Daniel S. Katz's Blog*. https://danielskatzblog.wordpress.com/2021/06/28/fair-is-not-the-end-goal/

Katz, D. S., Gruenpeter, M., & Honeyman, T. (2021). Taking a fresh look at FAIR for research software. *Patterns*, *2*(3), 100222. https://doi.org/10.1016/j.patter.2021.100222

Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E., Dominguez Del Angel, V., van de Sandt, S., Ison, J., Martinez, P. A., McQuilton, P., Valencia, A., Harrow, J., Psomopoulos, F., Gelpi, J. L., Chue Hong, N., Goble, C., & Capella-Gutierrez, S. (2020). Towards FAIR principles for research softwares. *Data Science*, *3*(1), 37–59. https://doi.org/10.3233/DS-190026

Funded by the European Union

Martinez-Ortiz, C., Kuzak, M., Spaaks, J. H., Maassen, J., & Bakker, T. (2020). *Five recommendations for "FAIR software."* https://doi.org/10.5281/zenodo.4310217

Martinez-Ortiz, C., Martinez Lavanchy, P., Sesink, L., Olivier, B. G., Meakin, J., de Jong, M., & Cruz, M. (2023). *Practical guide to Software Management Plans (1.1)*. Zenodo. https://doi.org/10.5281/zenodo.7589725

Möller, S., Prescott, S. W., Wirzenius, L., Reinholdtsen, P., Chapman, B., Prins, P., Soiland-Reyes, S., Klötzl, F., Bagnacani, A., Kalaš, M., Tille, A., & Crusoe, M. R. (2017). Robust Cross-Platform Workflows: How Technical and Scientific Communities Collaborate to Develop, Test and Share Best Practices for Data Analysis. *Data Science and Engineering*, *2*(3), 232–244. https://doi.org/10.1007/s41019-017-0050-4

Moltó, G., Campos, I., Hardt, M., Blanquer, I., Caballer, M., Orviz, P., David, M., Gomes, J., & EOSC-SYNERGY. (2020). *State of the art regarding digital badge issuing technologies*. https://doi.org/10.13039/501100000780

Morrissey, S. M. (2020). *Preserving Software: Motivations, Challenges and Approaches*. Digital Preservation Coalition. https://doi.org/10.7207/twgn20-02

NISO JATS4R Working Group, Software Citations Subgroup. (n.d.). *NISO JATS4R Software Citations v1.0*. NISO. https://doi.org/10.3789/niso-rp-40-2021

Nosek, B. (2019, June 11). Strategy for Culture Change. *Center for Open Science Blog*. https://www.cos.io/blog/strategy-for-culture-change

Oddou, A., & Henicz, M. (2023, April). *Protection of software under intellectual property law*. Inria training session, Lyon.

OpenAIRE, Baglioni, M., Bardi, A., Manghi, P., & Mack, L. (n.d.). *OpenAIRE Guidelines for Software Repository Managers—OpenAIRE Guidelines for Software Repository Managers 1.0 alpha documentation*. OpenAIRE. Retrieved April 6, 2023, from https://software-guidelines.readthedocs.io/en/latest/

Patel, B., Soundarajan, S., & Hu, Z. (2022). *Making Biomedical Research Software FAIR: Actionable Step-by-step Guidelines with a User-support Tool* (p. 2022.04.18.488694). bioRxiv. https://doi.org/10.1101/2022.04.18.488694

Peer, L., Green, A., & Stephenson, E. (2014). Committing to Data Quality Review. *International Journal of Digital Curation*, *9*(1), 263–291. https://doi.org/10.2218/ijdc.v9i1.317

Persistent uniform resource locator. (2023). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Persistent_uniform_resource_locator&oldid=1154787937

Raymond, E. S. (2000). *Software Release Practice HOWTO*. The Linux Documentation Project. https://tldp.org/HOWTO/html_single/Software-Release-Practice-HOWTO/

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

Romano, J. D., & Moore, J. H. (2020). Ten simple rules for writing a paper about scientific software. *PLOS Computational Biology*, *16*(11), e1008390. https://doi.org/10.1371/journal.pcbi.1008390

Science Code Manifesto. (2013). *Science Code Manifesto*. https://web.archive.org/web/20130502005551/http://sciencecodemanifesto.org/

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

Software Heritage. (2019). Save and reference research software. *Software Heritage*. https://www.softwareheritage.org/save-and-reference-research-software/

Software Heritage. (2020a, March 18). Software Heritage contributes CodeMeta generator to the community. *Software Heritage*. https://www.softwareheritage.org/2020/03/18/codemeta-community/

Software Heritage (Director). (2020b, July 7). *Tutorial: Citing software using biblatex-software*. Software Heritage. https://www.youtube.com/watch?v=UhQCeAj9yKM

Software Heritage (Director). (2022, January 27). *EOSC Software Infrastructures for Research Software: J. B. Gonzalez Lopez (CERN)*. https://www.youtube.com/watch?v=dc1fbSStYBw

Funded by the European Union

Software Heritage (Director). (2023, February 6). *Ethan Gates - EaaSI Access to Legacy Software: Designing Emulation Services for the Future*. https://www.youtube.com/watch?v=f6deYP9fvJU

Soiland-Reyes, S., Sefton, P., Crosas, M., Castro, L. J., Coppens, F., Fernández, J. M., Garijo, D., Grüning, B., La Rosa, M., Leo, S., Ó Carragáin, E., Portier, M., Trisovic, A., RO-Crate Community, Groth, P., & Goble, C. (2022). Packaging research artefacts with RO-Crate. *Data Science*, *5*(2), 97–138. https://doi.org/10.3233/DS-210053

Task Force on Best Practices for Software Registries, Monteil, A., Gonzalez-Beltran, A., Ioannidis, A., Allen, A., Lee, A., Bandrowski, A., Wilson, B. E., Mecum, B., Du, C. F., Robinson, C., Garijo, D., Katz, D. S., Long, D., Milliken, G., Ménager, H., Hausman, J., Spaaks, J. H., Fenlon, K., … Morrell, T. (2020). *Nine Best Practices for Research Software Registries and Repositories: A Concise Guide* (arXiv:2012.13117). arXiv. https://doi.org/10.48550/arXiv.2012.13117

The Carpentries. (n.d.). *Introduction to Workflows with Common Workflow Language*. The Carpentries. Retrieved April 6, 2023, from https://carpentries-incubator.github.io/cwl-novice-tutorial/

The Givaro Group. (2019). *Givaro* (4.1.1). https://archive.softwareheritage.org/swh:1:dir:df65912bd1e5ea4b96b935de95f6638eb6d9472d;origin=https://hal.archives-ouvertes.fr/hal-02130729;visit=swh:1:snp:da818bf8900aa772999896546db83b8e9cbe9f6f;anchor=swh:1:rev:5a63bd17a83030740eb27a980fe14a7a37f47efe;path=/. https://hal.science/hal-02130729

The Research Software Alliance. (2021, April 28). Guidelines. *The Research Software Alliance*. https://www.researchsoft.org/guidelines/

The Software Sustainability Institute. (2018). *Software Deposit Guidance for Researchers*. https://softwaresaved.github.io/software-deposit-guidance/

The Turing Way Community. (2022). *The Turing Way: A handbook for reproducible, ethical and collaborative research*. Zenodo. https://doi.org/10.5281/ZENODO.3233853

Turello, D. (2019, February 7). How to Think About Data: A Conversation with Christine Borgman [Webpage]. *Insights. Scholarly Work at the John W. Kluge Center*. https://blogs.loc.gov/kluge/2019/02/how-to-think-about-data-a-conversation-with-christine-borgman/

Vrandečić, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, *57*(10), 78–85. https://doi.org/10.1145/2629489

Würsch, M., Ghezzi, G., Hert, M., Reif, G., & Gall, H. C. (2012). SEON: A pyramid of ontologies for software evolution and its applications. *Computing*, *94*(11), 857–885. https://doi.org/10.1007/s00607-012-0204-1

Appendices

Appendix A: Task 4.3 description of work as submitted in the project proposal

T4.3 Standard metadata for research software [M1-M28] INRIA(12), UEDIN-SSI(4), UPM(2), DataCite (2), UKRI-STFC(2), UNIMAN(2). This task will develop guidelines for the collection and curation of metadata to archive, reference, describe and cite research software. We will survey the ecosystem of scholarly infrastructures and review existing guidelines in this area. We will follow recommendations from the EOSC SIRS report and actions from FAIR4RS, and evaluate how standard metadata impacts software reproducibility. With T4.4, this task will contribute to the normalisation efforts of CodeMeta and its crosswalk table to extend interoperability, as well as exploring possible synergies with initiatives such as FAIR Computational Workflow principles and community efforts like Bioschemas. Liasing with the EOSC 01-03 project, our guidelines will ease adoption of existing and new processes and tools in the scholarly ecosystem. With T5.2, we will feed into

Funded by the European Union

"FAIR metrics" for research software. With WP3 we will contribute to ISO standardization efforts of software identification using the SoftWare Heritage IDentifier (SWHID).

Appendix B: RSMD Checklist

*The Research Software MetaData (RSMD) guidelines are a cross-disciplinary community driven effort to provide standardized and actionable guidance for the metadata collection and curation of research software. To assist **software creators** in self-archiving and self-metadata curation, FAIR-IMPACT has created the RSMD checklist as a self contained questionnaire. The RSMD checklist which is addressing all the RSMD guidelines recommendations as a set of questions that can be used by the software creator or the software curator. By using the checklist, software creators can ensure that their software has the necessary metadata to be accessible, discoverable and reusable.*

General Metadata Requirements

- ☐ Does the software have metadata that is embedded in the source code (intrinsic metadata)? (RSMD-1.1)
- ☐ Does the software project have a metadata record (extrinsic metadata) which is publicly available on an online scholarly platform? (RSMD-1.2)
  - ☐ Is the metadata record licensed as CC0?
- ☐ Is the software also available in a version control system (VCS)? (RSMD-1.3)
  - ☐ if it is available online, is the url available in a code repository property in the intrinsic or/and extrinsic metadata?
- ☐ Does the software follow language specific community standards? (RSMD-1.4)
- ☐ Is the machine readable metadata information in a single file? (RSMD-1.5)

Accessibility and preservation

- ☐ Is the software record or/and are the software artifacts accessible and preserved?
  - ☐ Is the software source code preserved in the SWH universal source code archive, Software Heritage? (RSMD-2.1)
  - ☐ Is the software archived in a scholarly repository (e.g Zenodo, HAL)? (RSMD-2.2)
  - ☐ Can it be accessed and downloaded?
- ☐ Is the software registered in a disciplinary or community registry (e.g ascl.net, bio.tools, swMath, RRID portal, RSD, WikiData, DataCite)
  - ☐ Can it be found in a search engine?

Reference and identification

- ☐ Are the software versions clearly identified? Are the versions and contributions tracked? Using a VCS is preferable (RSMD-3.1)
- ☐ Are intrinsic identifiers available:
  - ☐ Can specific algorithms or code fragments be identified? (RSMD-3.2)
  - ☐ Can a file or directory be identified? (RSMD-3.2)
  - ☐ Can different revisions or releases be identified? (RSMD-3.2)
- ☐ Are extrinsic identifiers available:
  - ☐ Can different releases be specifically identified? (RSMD-3.3)
  - ☐ Can the project be identified? (RSMD-3.3)
  - ☐ If applicable, can a module be identified? (RSMD-3.3)
- ☐ Is a versioning scheme used? (RSMD-3.4)
- ☐ Is it possible to identify different levels of granularity? (RSMD-3.5)

Funded by the European Union

Description & classification

- ☐ Is the information about the software name and description available (on README file or other intrinsic metadata file)? (RSMD 4.1)
- ☐ Does the metadata record contain descriptive metadata? (RSMD 4.2)
  - ☐ Name
  - ☐ Description
  - ☐ Domain
  - ☐ Programming language
  - ☐ Date created
  - ☐ Date of first publication
  - ☐ Keywords
  - ☐ Related links
  - ☐ Version
- ☐ Is it possible to access articles describing the software via a persistent identifier, or at least, a stable URL? (RSMD 4.3)
- ☐ Is the intrinsic metadata of the software description available in a machine readable file? (RSMD 4.4)
- ☐ Is additional information (e.g. provenance information, functionalities, development status, etc.) on the software available in the README file? (RSMD 4.5)

Attribution & credit

- ☐ Are the authors & contributors identified and acknowledged?
  - ☐ Is the author's information available in the source code as intrinsic metadata? (RSMD-5.1)
  - ☐ Is the author's information available on the metadata record as extrinsic metadata? (RSMD-5.2)
  - ☐ Is the list of authors exhaustive or is a collective author used? (RSMD-5.8)
- ☐ Are people identifiers used? (ORCID, ID-HAL, ID-REF, etc.) (RSMD-5.3)
- ☐ Are the roles of the authors and contributors specified? (RSMD-5.4)
- ☐ Is a citation preference provided? (RSMD-5.5)
- ☐ If applicable, in the article citing the software, is the appropriate granularity used? (RSMD-5.6)

Reuse, licensing and legal aspects

- ☐ Before choosing a license, did you verify who is the rights holder of the software? (RSMD-6.1)
- ☐ Is the license information available in the source code (intrinsic metadata file)? (RSMD-6.2)
  - ☐ If there are several licenses, are all defined in the software source code?
  - ☐ In the source code are the following attributes available alongside the license: name of the software, year, copyright holder, contact information (email), license identifier (e.g. SPDX[45]) (RSMD-6.6)
- ☐ Is the license information available on the metadata record? (RSMD-6.3)
  - ☐ Is the license information in the code the same as the property on the metadata record?

[45] https://spdx.org/licenses/

Funded by the European Union

- ☐ Are external software modules used identified with their authors and license? for compliance purposes, it is necessary to verify licenses compatibility (RSMD-6.4)

Re-execute: Dependencies and execution environment

- ☐ Are the software dependencies described (RSMD-7.1)
  - ☐ Is the dependencies description available in a machine-actionable format?
- ☐ Is the operating system and relevant environment requirements described? (RSMD-7.2)
- ☐ Is the hardware and relevant hardware platform requirements described? (RSMD-7.3)
- ☐ Are the build instructions available (RSMD-7.4)
  - ☐ If applicable, are the build configuration files for the specific ecosystem (Makefile, Ant files, Dockerfile, etc.) and appropriate test cases provided?
- ☐ Is the user documentation available? (RSMD-7.5)
  - ☐ Are the input/output formats specified?
  - ☐ Are there examples available?
- ☐ Is the data used and/or produced by the software described or linked? (RSMD-7.6)
  - ☐ If applicable, is there a sample provenance trace/log of an execution, including pointers to input data and expected results available?

Funded by the European Union

Appendix C: Metadata Use Case Collection

Table I - Explanation of the actions required by the different stakeholders

Stakeholder	Action Needed	Goal of the Action
Researcher / Research Engineer / Developer (as creators of the software)	Archive software artifacts	Ensure long term accessibility and availability
	Reference software in articles	Ensure transparency and reproducibility of research
	Contribute and improve existing software	Collaborate with other developers and enhance software functionality
	Get credit for developed software	Facilitate career advancement and collaboration opportunities
	Identify copyright holders (of dependencies)	Ensure compliance with licensing and intellectual property rights
	Cite software and curate citation entries	Properly reference and acknowledge software contributions
	Verify/reproduce/improve results	Track software versions and dependencies for result replication and improvement
	Ensure compatibility with environment	Prevent incompatibilities with software dependencies
	Compare performance of different operating systems	Make informed decisions regarding software implementation
Laboratory/team	Track software contributions	Document and collaborate on software development within the organization
	Produce reports on software	Showcase technological capabilities and achievements
	Maintain web page/repository	Provide easy access and visibility to software assets

Funded by the European Union

Stakeholder	Action Needed	Goal of the Action
Curators and librarians, digital archivists	Curate software metadata	Ensure accurate and comprehensive software descriptions
	Provide documentation on software preservation	Disseminate good practices for long-term software sustainability
	Monitor research teams' production	Capture and organize software outputs effectively
Users	Search for relevant software	Find tools that assist in data analysis or other research activities
	Cite creator of the code	Provide proper attribution in citations and acknowledgments
	Refer to a specific software version	Accurately cite software when reproducing older research
	Determine permissible use for research	Assess the software's suitability for research purposes
	Illustrate problem-solving using specific software	Enhance educational experiences with practical examples
	Check the modified date	Determine if the software is up to date and actively maintained
	Compile and execute research software	To utilize functionalities and obtain meaningful results

Funded by the European Union

Appendix D: The Difference Between Intrinsic And Extrinsic Identifiers

Table II - Overview of the main differences between intrinsic and extrinsic identifiers

	Intrinsic identifiers	***Extrinsic identifiers***
Goal	The focus is on **reproducibility**: Reuse a software | Verify, Improve results linked to the software	Describe a software - including attribute credit to the authors or creators
Use case	Retrieve byte-identical copies of source code artifacts	Refer to a given software in a catalog, a citation | Track the software produced by an institution
Research object type[46]	Software artifacts = digital objects produced during the development stages, such as a repository, a release, a directory, a single file, a commit, etc. but also, executables	Software project = a concept, an abstract entity. "an endeavor to develop and maintain software artifacts" (Di Cosmo et al., 2020)
Example of object	Excerpt of source code[47]	Parmap, Scikit-Learn, Coq, etc.
Type of PID	Identifiers for Digital Objects	Digital Identifiers of Objects
Technical challenges	Granularity of the object | Software lifecycle | Economic sustainability[48]	Traceability, identification of all the contributors: the list of authors and their specific role during the different stages of the development may evolve regularly. (Alliez et al., 2019; Canteaut et al., 2021) Multiple vocabularies to describe softwares (see MD Analysis: Software vocabularies and ontologies landscape)
Challenges faced by end-users	Identify which PID fits to which need | Adopt the use of forges for software development	Lack of standardized citation practices for software | Understanding the different types of contributions (e.g. architecture, design, test, documentation, etc.)
Stakeholders	Depends on the use case, but mainly for: ● Researchers ● Software developers ● Research teams	Depends on the use case, but mainly for: ● Research institutions ● Scholarly repositories ● Publishers ● Aggregators

[46] *For further information, see the diagram below : (*Research Data Alliance/FORCE11 Software Source Code Identification WG)

[47] swh:1:cnt:bb0faf6919fc60636b2696f32ec9b3c2adb247fe

[48] "In the case of digital resources that need to be created or modified frequently, and especially when their amount is very large, charging as per identifier fee is problematic."

Funded by the European Union

Appendix E: Metadata Types

Table III - Overview of the core specificities of intrinsic and extrinsic metadata

	Intrinsic Metadata (MD)	*Extrinsic Metadata (MD)*
***What** is this type of MD?*	Embedded within the content or object itself, which can be automatically extracted or inferred from the content	As opposed to intrinsic MD, it is the MD record describing the software externally to the source code.
***Where** is the MD found?*	In text files inside the source code directory, using standard name scheme depending on the metadata type.	On a platform separately from the code: - Development platform (e.g forge), - Scholarly repository - Registry, - Publishers interface - Package manager record. - Website page
***Who** provides/curates the MD?*	Often created and managed by the software creators.	Assigned and maintained by external entities such as aggregators, registries, publishers, or scholarly repositories. Can also include descriptive metadata that is manually created or curated by humans, such as related research outputs.
***How** is MD managed?*	Managed by the software creators. The content of the MD files can be automatically extracted or inferred from the content.	The software record can be created by the software authors or by another stakeholder who needs a MD record. It is managed by the MD record creator.
***Why** should we need to preserve this MD?* *Which type of information is provided?*	Can be critical for preserving the content's authenticity and integrity	Can help to provide additional context or understanding of the software. May contain provenance information that can help users understand the software and its relation to other research outputs.
How is the MD preserved?	If the code is preserved then the metadata is preserved alongside the code.	MD is preserved only if the platforms deposit the metadata into an archive.

Funded by the European Union

Appendix F: Description of Infrastructures Types

Table IV - Definition and examples of existing infrastructures and platform types

Infrastructure/ Platform type	***Definition/ Why do we focus on this actor/infrastructure?***	***Examples***
Scholarly Repositories	"An organisation called to archive and make available research artifacts, e.g. articles, datasets, software." (EOSC Executive Board & EOSC Secretariat, 2020)	• HAL • Zenodo • Dryad
Registries (catalogs)	"*Research software registries* are typically indexes or catalogs of software metadata, without any code stored in them; while in *research software repositories*, software is both indexed *and* stored (Lamprecht et al., 2020)." (Garijo et al., 2022)	• The DataCite Metadata collection • swMATH • OpenAire
Publishers	"Any organization that prepares submitted research texts, possibly with associated source code and data, to produce a publication and manage its dissemination, promotion, and archival process." (EOSC Executive Board & EOSC Secretariat, 2020) "[…] there is an opportunity for publishers to educate authors on the necessity of sharing software source code and encourage a standard workflow." (EOSC Executive Board & EOSC Secretariat, 2020)	• Dagstuhl • IPol
Aggregators	"Aggregators collect, curate, select, present, and aggregate information about research software from various sources to improve findability in diverse communities."(EOSC Executive Board & EOSC Secretariat, 2020) "Any service that collects information about digital content from a variety of sources with the primary goal of increasing its discoverability, and possibly adding value to this information via processes like curation, abstraction, and classification, and linking."(EOSC Executive Board & EOSC Secretariat, 2020)	• OpenAIRE • swMATH.org
Software development platform	An online service for developers to collaborate on software development activities https://en.wikipedia.org/wiki/Collaborative_development_environment https://en.wikipedia.org/wiki/Version_control Note this infrastructure is usually outside the academic realm.	• GitHub • Bitbucket • SourceForge
Package managers	A repository of software tools and libraries that can be installed on a given base system or for a particular programming language; typically as intra-dependent packages with pre-compiled binaries or build recipes. "The foundations of functional workflows are sources for readily usable software" https://doi.org/10.1007/s41019-017-0050-4	• OS: Debian, Ubuntu, WinGet • OS independent: Conda, Homebrew • Python: PIP • R: CRAN

Funded by the European Union

Appendix G: Status of CodeMeta Adoption In Various Infrastructures

Table V – Existing Infrastructures using CodeMeta or engaged in development to provide CodeMeta import, export or interoperability functionalities

Infrastructures	**type**	**import format**	**export format**	**interoperability with other infra**	**Planned release with CodeMeta support**
ASCL; Astrophysics Source Code Library	aggregator	TBC	yes		supported
swMATH	aggregator	not planned	planned	SWH	June 2025
OpenAire	aggregator		planned		June 2025
SWH	archive	yes - codemeta.json indexers and translator of intrinsic metadata	planned	All git, svn, hg version control systems & many package managers	supported
HAL	scholarly repository	yes	yes	SWH	supported
DataVerse	scholarly repository				
Zenodo.org with InvenioRDM	scholarly repository	zenodo.json	planned		June 2025
IPOL	publisher	codemeta	planned		
eLife	publisher				
Dagstuhl	publisher		planned	SWH	June 2025
Episciences - JTCAM	publisher		planned		June 2025

Funded by the European Union

Appendix H: The Metadata Landscape: Software Ontologies and Vocabularies

The metadata landscape encompasses many software ontologies, vocabularies, and metadata schemes designed to capture and represent essential information about software artifacts, and tailored for discovery.
These frameworks provide structured and standardized approaches to capture essential information about software, ranging from its version and configuration to input/output specifications and beyond.

Dedicated software ontologies

Description of a Project[49] (DOAP**)**, Asset Description Metadata Schema for Software[50] (ADMS.SW), and the family of Software Evolution ONtologies (SEON[51]) (Würsch et al., 2012), provide detailed definitions of software artifacts with an emphasis on FOSS. Ontologies like **SD**[52] and **SDM**[53] focus on defining different software versions and configurations and constraints, while vocabularies like **SWO**[54] define types of inputs and outputs in the biomedical domain.
DOAP, ADMS.SW and SEON try to tackle the problem of what is software from a different perspective. The ADMS.SW, a European Commission project started in 2011, is now considered as deprecated. Nevertheless, it is used on the Adullact forge. Its specifications are exhaustive and show a detailed definition of software artifacts with an emphasis on FOSS. SEON has a different point of view on the evolution of software that is rarely found in the metadata schemes landscape (Würsch et al., 2012). DOAP keeps it simple, while ADMS.SW and SEON are more complex. One of DOAP advantages is that it aims on offering interoperability with other popular Web metadata projects (RSS, FOAF, Dublin Core). DOAP is used on Freecode, SourceForge and other software catalogs and was used as a base to create the ADMS.SW specification.
The Software Description Ontology builds on Codemeta and Schema.org to represent different aspects of software (i.e. versions, configurations and input/outputs) to capture execution details and facilitate interchanging data between different software components. Ontologies like the Software Description Ontology for Models and SWO focus on specific domains (Geoscience models and Biomedical domain, respectively). For example, SWO reuses EDAM to describe the types of inputs and outputs of biomedical software.
The EDAM Ontology (Ison et al., 2013) is used to add bioinformatics-specific metadata, such as strong typing of inputs and outputs, within Abstract CWL and Bioschemas annotations. EDAM is also used to describe the overall workflow Topics and Operations (requested by users) to help find workflows. It is possible to search and filter using these properties. Note that the approach is not limited to EDAM, e.g. in terms of attribute names. This supports expansion beyond EDAM in the future to support non-life-sciences - although EDAM themselves are also currently going in that direction - topics are starting to support ecology, mathematics, language, physics, chemistry plus many more non life science terms.

[49] https://github.com/ewilderj/doap/wiki
[50] https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/asset-description-metadata-schema-software/release/v100
[51] http://se-on.org/
[52] https://w3id.org/okn/o/sd/
[53] https://w3id.org/okn/o/sdm/
[54] https://www.ebi.ac.uk/ols/ontologies/swo

Funded by the European Union

Linked data schemes

From a semantic web point of view, software should be referenced via linked data to facilitate its discoverability on the web. **schema.org** and **Wikidata**, enable rich search results on various search engines by providing software-related entities. For example, schema.org, sponsored by search engines, promotes a very pragmatic approach to the semantic web, by providing these software-related entities: SoftwareApplication and SoftwareSourceCode. Wikidata, the free and open knowledge base that can be read and edited by both humans and machines (Vrandečić & Krötzsch, 2014), is a great place to extract or query metadata.

Scholarly ecosystem

OntoSoft is an ontology with many properties adapted to scientific software in the academic domain (Gil et al., 2015) that supports particular use cases, giving credit via citations. **TOTEM** (Trustworthy Online Technical Environment Metadata Database) for digital objects in general also created in and for the scholarly ecosystem. **DataCite**'s metadata schema is used by Zenodo and is specified for research artifacts, including data and software. The **Citation File Format**[55] (CFF) was introduced in 2017 with the first release candidate of the specs. It provides a simple way to capture research software citation information in a code repository.

Digital preservation schemes

These schemes are used in a variety of domains far from the engineering scene and don't have software specific properties. To name a few, we have:

- Dublin Core with 15 standardized properties that can be used with other metadata schemes (DOAP for example)
- Marc :MAchine-Readable Cataloguing (MIT is using MARC records),
- PRONOM is a technical registry for digital artifacts including software at the UK National Archives using its own specifications and a unique id called PUID.
- PREMIS- Preservation Metadata Maintenance Activity, also an international standard developed for the Library of Congress, allows the usage of different vocabulary if the vocabulary is specified.

Package management metadata management files

These file formats are used mostly in the source code itself as manifest files. Each package manager system provides a filename and format to share metadata, for example: for Java-Maven projects, use `pom.xml`; for Ruby gems, use either `.gemspec` or `Rakefile`; for R packages, use `DESCRIPTION`; for JavaScript npm packages, use `package.json`; for PyPI packages, use `pyproject.toml` or `setup.cfg`.

[55] https://citation-file-format.github.io/

Appendix I: Domain Specific & Community Driven Related Efforts

In this appendix, we delve into a range of domain-specific and community-driven efforts that are dedicated to addressing the challenges and promoting best practices for Research Software. These initiatives and guidelines aim to improve metadata, workflows, and tool descriptions within specific domains and communities, ultimately fostering reproducibility, collaboration, and the adoption of FAIR principles.

FAIR Biomedical Research Software guidelines (FAIR-BioRS)

Effort from the Biomedical community[56] to propose actionable guidelines for research scientists. The guidelines (Patel et al., 2022) include a call for including concrete metadata in a repository, using a CFF file or the CodeMeta generator:
"Provide, at least, the following fields in the codemeta.json file: Software name ("name"), Software description/abstract ("description"), Unique identifier ("identifier"), Authors ("givenName", "familyName") with their Organization name ("affiliation"), Keywords ("keywords"), Programming Language ("programmingLanguage"), First and current release date ("dataPublished" and "dateModified"), License used ("license").

Model Cards

With the popularity of Machine Learning, public platforms like HuggingFace[57] have been developed to help the community share and reuse trained models. However, there are no standard metadata vocabularies to describe Machine Learning models. Model Cards[58] have been proposed as a means to describe the main features of a model, as well as important features in their training process. Although no metadata fields are formally described, model cards encourage "language", "tags", "license" as well as dataset features used in training (dataset source links) and validation metrics. HuggingFace implements model cards[59] providing guidelines on how to describe a model .

Computational Workflows in the Bioinformatics Community

Many disciplines have adopted computational workflows as a software framework that handle the complexity of linking data flows across different codes and executing them on different computing platforms. There is a drive in the bioinformatics community towards adopting FAIR practices for workflows, taken up by organizations running tool registries (ELIXIR's bio.tools) and workflow services (e.g. ELIXIR's WorkflowHub[60] and workflow managers, DockStore, KNIMEhub etc) and workflow managers and community specific repositories (e.g. Galaxy and IWC; Nextflow and nf-core etc). The Workflow Community Initiative[61] FAIR Computational Workflows WG has convened to define best practices and principles for FAIR workflows, included in WCI's 2022 roadmap[62].
As proposed by Goble et al (Goble et al., 2020) workflows are hybrid processual objects with a tight coupling with their data; their workflow language description abstraction can be subject to FAIR data principles but they are also a form of executable compositional objects, as well as their execution managers being software.

[56] https://github.com/FAIR-BioRS/Guidelines
[57] https://huggingface.co/
[58] https://research.google/pubs/pub48120/
[59] https://huggingface.co/docs/hub/model-cards
[60] https://workflowhub.eu/
[61] https://workflows.community/about
[62] https://arxiv.org/abs/2304.00019

Funded by the European Union

In the EOSC-Life bioscience's workflow colaboratory, ELIXIR, the research infrastructure for life science data and BBMRI-ERIC, the research infrastructure for biobanking, led the development and implementation of a metadata framework for computational workflows.

Bioinformatics Standards

Bioschemas[63] schema.org profiles for Computational Tool[64], Computational Workflow[65] and Formal Parameter[66] provide metadata about a workflow and its tools that are discipline independent. All include properties @context, @type, @id, and dct:conformsTo used to state the Bioschemas profile that the markup relates to. The versioned URL of the profile must be used

- Computational Workflow[67] has 10 mandatory fields (creator, dateCreated, license, name, url, input, output, programmingLanguage, sdPublisher and version), 15 recommended and 7 optional.
- Formal Parameter[68] has 1 mandatory field (name), 3 recommended and 3 optional.
- Computational Tool[69] has 3 mandatory fields (description, name, url), 7 recommended and 20 optional. This profile is used to mark up tools in the ELIXIR bio.tools registry.

Despite the prefix "Bio" BioSchemas for tools, workflows and formal parameters has nothing specific to biology and is domain agnostic. Like Bioschemas, CodeMeta is a list of properties relevant for research software, but unlike Bioschemas it does not have priority recommendation (minimum, recommended, optional) or recommendation for entity typing (SoftwareApplication, SoftwareSourceCode). 73% of Bioschemas's ComputationalTools properties are already in CodeMeta.

The Common Workflow Language (Crusoe et al., 2022) is a canonical workflow description to accompany native workflow definitions – both *CWL workflows* (which are executable) and *Abstract CWL* (non-executable) description variants. This description presents the structure of workflows, composed of tools and external interface, in an interoperable way across workflow languages.

RO-Crate (Soiland-Reyes et al., 2022), a community-developed standardized approach for research output packaging with rich metadata. RO-Crate provides us with the ability to package executable workflows, their components, such as example and test data, abstract CWL, diagrams and their documentation. This makes workflows more readily re-usable. RO-Crate is based on schema.org, and is aligned with CodeMeta, however with a more general focus as the software description is not required. RO-Crate is the base unit of circulation between services such as WorkflowHub, LifeMonitor and workflow managers like Galaxy Europe.

Three RO-Crate profiles have been developed for Workflows:

- Workflow-RO-Crate[70] which WorkflowHub can register and mint; considers the workflow as a software element

[63] https://bioschemas.org
[64] https://bioschemas.org/profiles/ComputationalTool/1.0-RELEASE
[65] https://bioschemas.org/profiles/ComputationalWorkflow/1.0-RELEASE
[66] https://bioschemas.org/profiles/FormalParameter/1.0-RELEASE
[67] https://bioschemas.org/profiles/ComputationalWorkflow/1.0-RELEASE
[68] https://bioschemas.org/profiles/FormalParameter/1.0-RELEASE
[69] https://bioschemas.org/profiles/ComputationalTool/1.0-RELEASE
[70] https://about.workflowhub.eu/Workflow-RO-Crate/

Funded by the European Union

- Workflow-Testing-RO-Crate[71] which includes test information for LifeMonitor;
- Workflow-Run-RO-Crate[72] which includes provenance information after the execution of a workflow or other computational tools.

Identifiers for all the components: RO-Crates can be metadata-rich bags of identifiers and can themselves be assigned permanent identifiers. This enables the full description of a computational analysis, from input data, over tools and workflows, to final results.

Genomics and Health

The metadata framework and Global Alliance for Genomics and Health APIs enable interoperability with the services of the Collaboratory. The Tools Registry Service API[73] supports the exchange of scientific tools and workflows and enables users to search for and retrieve metadata about registered tools, including the tool's name, version, description, author, input and output parameters, and Docker image details. citation.cff[74] placed in Git repositories is harvested by WorkflowHub. A set of recommendations for research software to be better integrated by workflow systems have been developed (Brack et al., 2022), e.g. *Make your tool parallelizable* by not relying on overwriting fixed file paths.

[71] https://crs4.github.io/life_monitor/workflow_testing_ro_crate
[72] https://www.researchobject.org/workflow-run-crate/
[73] https://ga4gh.github.io/tool-registry-service-schemas/
[74] https://citation-file-format.github.io/

Funded by the European Union

Appendix J: Community contributions from the FAIR-IMPACT webinar

Table VI - List of contributors that have participated in the Developing Guidelines for Metadata Collection and Curation for Research Software on May 23rd 2023.
During the webinar, the participants were invited to comment v1.0 of the RSMD guidelines.

Your last name	Your first name	Orcid	Institution	Country	Job title	I want to be listed as a contributor Y/N
Järveläinen	Pekka	0000-0003-3111-1515	CSC	Finland	Programmer	Y
Brinkman	Loek	https://orcid.org/0000-0003-3997-1173	DANS	The Netherlands	Research Data Specialist	Y
Fouilloux	Anne	https://orcid.org/0000-0002-1784-2920	Simula Research Laboratory	Norway	Research Engineer	Y
Juty	Nick	https://orcid.org/0000-0002-2036-8350	The University of Manchester	UK	Senior technical research manager	y
Löbe	Matthias	0000-0002-2344-0426	IMISE U Leipzig	DE	Researcher	Y
Himpe	Christian	0000-0003-2194-6754	University of Muenster	Germany	Data Engineer	Y
Davidson	Joy	0000-0003-3484-7675	DCC	UK	Associate Director	Y
Lesnikova	Angelina	https://orcid.org/0000-0003-4163-0044	sci2sci	Germany	CEO	Y
Vials Moore	Adam	0000-0002-2085-1908	Jisc	UK	Product Specialist - PIDs	Y
Solanki	Dhwani		ZB MED & Bonn University	Germany	Student research assistant	Y
Karatas	Tugce	https://orcid.org/0000-0002-9446-4679	C2DH / Unilu	Luxembourg	Research Data Manager	Y
Galkin	Anastasia	https://orcid.org/0000-00	AIP Potsdam	Germany	Research Infrastructure	Y

Funded by the European Union

Your last name	Your first name	Orcid	Institution	Country	Job title	I want to be listed as a contributor Y/N
		03-0131-7491			Specialist	
Streicher	Ole	https://orcid.org/0000-0001-7751-1843	AIP Potsdam	Germany	Support Scientist	Y
Andersen	Peter Bruhn		Agency for Digital Government	Denmark	Link Data Architect	y
Raga Raga	Núria	0000-0001-9643-6826	CSUC	Spain	Information Resources Specialist	Y
Rodani	Tommaso	0000-0003-0570-3509	AREA Science Park	Italy	PhD Student	Y
Dhollander	Evelien	0000-0001-6302-2897	Ghent University	Belgium	Data Curator	Y
Claudia	Fratianni	0000-0002-4983-4255	INGV	Italy	Technologist	Y
Andreou	Panayiotis	0000-0002-6369-1094	UCLan Cyprus	Cyprus	Associate Professor	Y
Nivault	Estelle	0000-0003-0630-5633	Inria	France	Librarian	Y
Domhnall	Carlin	0000-0002-8424-2757	Queen's University Belfast	UK	Research Software Engineering Fellow	Y
Azzouz-Thuderoz	Maxence	0000-0002-8710-9548	FIZ Karlsruhe	Germany	Scientist	Y

Funded by the European Union