

Project Number: 101057264

Start Date of Project: 01/06/2022

Duration: 36 months

Deliverable D6.1

Report on Standardisation and Curation of Software Metadata and PIDs

Dissemination Level	PU
Due Date of Deliverable	(30/09/24), Month 28
Actual Submission Date	(30/09/2024)
Work Package	WP6 - Services and tools to archive, reference, describe and cite research software
Task	T6.4 - Metadata and PIDs for Software: Curation and Standardisation
Type	Report
Version	V 1.0
Number of Pages	p.1 – p. 51

Deliverable Abstract

The FAIRCORE4EOSC report on standardisation and curation of software metadata and PIDs offers an implementation overview and roadmap within the European Open Science Cloud (EOSC). It focuses on standardised practices and effective curation mechanisms, recognizing research software as a primary output. The report covers cultural changes, the role of software in scholarly communications, and provides definitions for standardisation and curation. It proposes an operational roadmap for adopting recommendations and improving interoperability. The document details curation workflows, quality maintenance strategies, and tools for metadata management. It also highlights standardisation efforts and community contributions, with a plan to improve interoperability and reduce redundancies.

DELIVERY SLIP

Contribution	Name	Partner/Activity	ORCID
Lead author(s)	M. Gruenpeter	INRIA	0000-0002-9777-5560
Contributor(s)	S. Granger	INRIA	0000-0002-9081-3851
	A. Monteil	INRIA	0000-0003-3150-4837
	J. Sadowska	INRIA	0000-0002-5193-9435
	E. Nivault	INRIA	0000-0003-0630-5633
	A. Ioannidis	CERN	0000-0002-5082-6404
	M.Azzouz-Thuderoz	FIZ	0000-0002-8710-9548
	M. Wagner	LZI	
	S. Bozaci	LZI	0009-0002-2202-4647
	S. Wimalaratne	DATA CITE	0000-0002-5355-2576
	S. Bingert	GWDG	0000-0001-9547-1582
Reviewer(s)	G. Cakir	GWDG	
	C. Goble	UNIMAN (EVERSE)	0000-0003-1219-2137
	D. Garijo	UPM (EVERSE)	0000-0003-0454-7145
	J. Maassen	eScience Center (EVERSE)	0000-0002-8172-4865
Editorial review	E. Halonen	CSC	

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided "as is" without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability. This deliverable is licensed under a Creative Commons Attribution 4.0 International License.



DOCUMENT LOG

Issue	Date	Comment	Author/Editor/ Reviewer
v.0.1	03-05-2024	Inception and first draft	M. Gruenpeter
v.0.2	15.06.2024	Contributions from T6.4 members	M. Gruenpeter & T6.4 co-authors
v.0.3	10.07.2024	Contributions from WP6	M. Gruenpeter & all co-authors
v 0.4	27.08.2024	First roadmap draft	M. Gruenpeter & all co-authors
v 0.5	02.09.2024	First version for external review	M. Gruenpeter & all co-authors



FAIRCORE4EOSC has received funding from the EU's Horizon Europe research and innovation programme under Grant Agreement no. 101057264.

v0.6	17.09.2024	Review by EVERSE partners	C.Goble, D.Garijo & J. Massen review
v0.7	24.09.2024	Integration of review	M. Gruenpeter & all co-authors
v0.8	30.09.2024	Final editorial review	E. Halonen final review
1.0	30.09.2024	Submitted report to EC	E. Halonen

TERMINOLOGY

Terminology/Acronym	Definition
ARDC	Archive, Reference, Describe & Cite
CFF	Citation File Format
EOSC	The European Open Science Cloud
EOSC PID Policy	The policy being developed by the EOSC PID Policy Task Force to ensure a minimum standard of performance for the PID ecosystem in EOSC
FAIR	Principles based on community expectations in respect of research outputs - findable, accessible, interoperable, and reusable.
FAIR-IMPACT	A EU-funded project that has as its main objectives to identify practices, policies, tools and technical specifications to guide researchers, repository managers, research performing organizations, policy makers and citizen scientists towards a FAIR data management cycle. The focus will be on persistent identifiers (PIDs), metadata, ontologies, metrics, certification and interoperability.
FAIRCORE4EOSC	The FAIRCORE4EOSC project focuses on the development and realization of core components for the European Open Science Cloud (EOSC).
JATS	Journal Article Tag Suite: XML format used to describe scientific literature published online
PID	Persistent identifier: generally expected to be unique, resolvable, and persistent.
RSAC	Research Software APIs and Connectors (EOSC)
RSMD	Research Software MetaData (guidelines)
SIRS	Scholarly infrastructures for research software
SPDX	System Package Data Exchange is an open standard capable of representing systems with digital components as bills of materials.
SWHID	SoftWare Hash Identifiers are designed to identify permanently and intrinsically all the levels of granularity that correspond to concrete software artifacts: snapshots, releases, commits, directories, files and code fragments.
SWORD	Simple Web-service Offering Repository Deposit is an interoperability standard developed by JISC extending ATOM.

Table of Contents

EXECUTIVE SUMMARY.....	6
1 INTRODUCTION.....	7
1.1 A cultural change: Research Software as a first-class output.....	7
1.1.1 Standardisation and curation definitions.....	8
1.2 EOSC Scholarly Infrastructures for Research Software recommendations.....	9
1.2.1 Summary of curation recommendations and gaps.....	10
1.2.2 Standardisation and curation implementation plan in FAIRCORE4EOSC.....	11
1.3 Who is the intended audience of this document?.....	11
2 STANDARDISATION.....	12
2.1 The CodeMeta metadata journey.....	12
2.1.1 What is CodeMeta?.....	12
2.1.2 The CodeMeta Community.....	12
2.1.3 A collaborative governance model.....	13
2.1.4 From CodeMeta v2.0 to CodeMeta v3.0.....	14
2.2 Identifying source code: The Software Hash Identifier.....	14
2.2.1 SWHID Working group and governance.....	16
2.2.2 Specifications and ISO submission.....	17
2.2.3 SWHID adopters.....	18
2.2.4 SWHID tools.....	19
2.3 Community.....	20
2.3.1 EOSC projects & task forces.....	20
2.3.2 International community driven initiatives.....	22
3 CURATION WORKFLOWS.....	24
3.1 State of the art of curation workflows.....	24
3.1.1 Sharing and self-archival.....	26
3.1.2 Dissemination and metadata curation or moderation: the HAL story.....	26
3.1.3 Publishing and peer review.....	29
3.2 Curation quality – keeping humans in the loop.....	31
3.2.1 Good enough metadata curation: sharing the cost of the metadata management.....	31
3.2.2 Ensuring metadata quality: the quest for more standardized practices.....	35
4 IMPLEMENTATION.....	37
4.1 Infrastructures: implementation of curation mechanisms.....	37
4.1.1 Scholarly repositories.....	37
4.1.2 Publishers.....	38
4.1.3 Aggregators.....	39

4.2 Metadata management and curation.....	41
4.2.1 Metadata landscape overview.....	41
4.2.2 Controlled vocabularies.....	42
4.2.3 Tools for metadata curation.....	44
5 THE CURATION & STANDARDISATION ROADMAP.....	48
5.1 Action steps for moving forward within EOSC and beyond.....	48
5.1.1 The challenges.....	48
5.1.2 Concrete steps for the EOSC community.....	48
5.1.3 Call to action.....	51
REFERENCES.....	52
APPENDICES.....	54
<i>Appendix A: SIRS Gap report analysis and actions.....</i>	<i>54</i>
<i>Appendix B: Description of Infrastructures Types from FAIR-IMPACT D4.4.....</i>	<i>57</i>
<i>Appendix C: Infrastructure metadata curation capabilities evaluation.....</i>	<i>58</i>
<i>Appendix D: Intrinsic vs. Extrinsic identifiers from FAIR-IMPACT D4.4.....</i>	<i>59</i>
<i>Appendix E: The Episciences platform.....</i>	<i>61</i>
<i>Appendix F: Metadata tools in the Research Software landscape.....</i>	<i>63</i>

List of Figures

Figure 1 - Pyramid from Strategy for Culture Change: Brian Nosek (2019).....	7
Figure 2 - Architecture of interconnected scholarly infrastructures supporting archival, reference, description and citation of (research) software source code.....	10
Figure 3 - Deposit workflow using a SWHID on scholarly infrastructure.....	28
Figure 4 - Extraction automation of metadata using a SWHID on HAL.....	29
Figure 5 - IPOL journal metadata record - published after peer review.....	30
Figure 6 - Figure 6: Curation workflow on HAL.....	32
Figure 7 - The Software Ontology (SWO) landscape (Malone et al., 2014).....	42
Figure 8 - CodeMeta generator actions.....	45
Figure 9 - The new Review box.....	45
Figure 10 - The new Role functionality.....	46
Figure 11 - Metadata workflow in Software Heritage following the metadata architecture.....	47
Figure 12 - Episciences platform.....	62

List of Tables

Table 1 - Intrinsic vs. Extrinsic metadata.....	12
Table 2 - Intrinsic vs. Extrinsic identifiers (extended in Appendix D).....	15
Table 3 - SWHID Working Group roles.....	16

Table 4 - SWHID specifications development process.....	17
Table 5 - Dissemination modalities in metadata curation workflows.....	24
Table 6 - fields in software records on HAL.....	33
Table 7 - SIRS Gap report analysis and actions.....	54
Table 8 - Definition and examples of existing infrastructures and platform types.....	57
Table 9 - Overview of the main differences between intrinsic and extrinsic identifiers.....	59
Table 10 - Comparison of metadata tools for Research Software.....	63

EXECUTIVE SUMMARY

The FAIRCORE4EOSC report on standardisation and curation of software metadata and PIDs is an implementation overview and roadmap of the efforts achieved by infrastructures in the evolving European landscape of research software. The report explores cultural changes in the recognition of research software as a first class output as recommended in the EOSC SRIA (Directorate-General for Research and Innovation (European Commission) & EOSC Executive Board, 2022) and in the EOSC SIRS report (EOSC Executive Board & EOSC Secretariat, 2020). It outlines the critical role of software in scholarly communications and provides definitions for standardisation and curation, emphasizing the need for standardised practices and effective curation mechanisms within the European Open Science Cloud (EOSC).

This report, D6.1, includes a proposal of an operational roadmap towards adoption of the existing recommendations and concrete steps to strengthen the interoperability between infrastructures using PIDs and metadata. Illustrating existing mechanisms and additional implementation by the FAIRCORE4EOSC Work Package 6: *Services and tools to archive, reference, describe and cite research software* and demonstrated by the Task 4.3 participants, Metadata and PIDs for Software: Curation and Standardisation, led by Inria. FAIRCORE4EOSC aims to develop EOSC-Core components to enable a FAIR (Findable, Accessible, Interoperable, and Reusable) European Open Science Cloud (EOSC) Ecosystem by addressing gaps identified in the Strategic Research and Innovation Agenda (SRIA).

The document is aimed at stakeholders involved in research software dissemination at the infrastructure layer, showcasing curation workflows such as sharing, self-archival, dissemination, metadata moderation, publishing, and peer review. It emphasizes the importance of maintaining curation quality through human intervention and defining minimum metadata quality standards, specifying possible pathways. Additionally, it details strategies for nurturing metadata moderators and supporting them with appropriate tools and training.

The standardisation efforts include the CodeMeta metadata governance journey, the Software Hash Identifier (SWHID), and contributions from community initiatives like the FAIR-IMPACT RSMD guidelines and the SciCodes consortium.

The implementation section covers various use cases and scenarios across different types of infrastructures including: archives, scholarly repositories, publishers, and aggregators, with a focus on metadata management and curation tools.

Finally, a roadmap outlines action steps for advancing curation and standardisation within EOSC and beyond. It addresses challenges and proposes a comprehensive plan to improve interoperability and reduce redundancies.

1 INTRODUCTION

1.1 A cultural change: Research Software as a first-class output

Software is essential for academia, playing a crucial role in research, while it is starting to be recognized, as seen in the [UNESCO recommendations](#)¹, the [DORA declaration](#)² and the [French national plan for Open Science](#)³, it is not yet fully recognized as a first-class output in many institutions. To move forward towards this culture change, where software is on par alongside publications and data, we can put effort at different levels, as we can identify in the Nosek pyramid on culture change below.

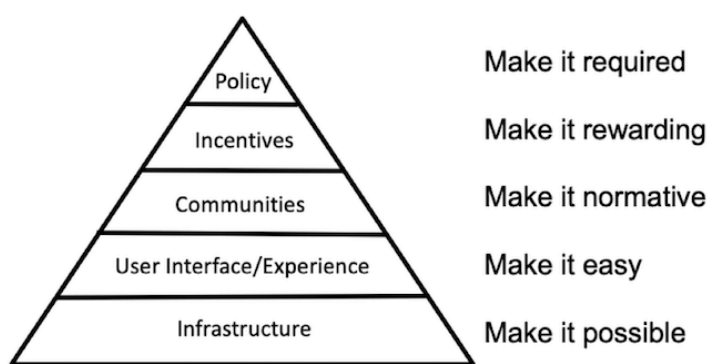


Figure 1 - Pyramid from Strategy for Culture Change: Brian Nosek (2019) <https://www.cos.io/blog/strategy-for-culture-change>

The base layer in the Nosek pyramid identifies infrastructures as the foundation for enabling culture change; without it, any other desire for change is impossible. We need to keep this in mind when moving forward and proposing incentives or requirements. If the infrastructures aren't in place the other layers can be ineffective and unsustainable. Infrastructures include **technical** and **people infrastructures**, including the networks, communities, and organizations that provide the human resources and expertise necessary to support and maintain scientific, technological, and research activities.

Quality metadata and standardized identifiers are only possible if the platforms and infrastructures provide users with the necessary functionalities. In this document we layout the status of software metadata and PIDs, the ongoing implementation in the FAIRCORE4EOSC infrastructures, including:

¹ <https://www.unesco.org/en/open-science/about>

² <https://sfdora.org/read/>

³ <https://www.ouvrirlascience.fr/second-national-plan-for-open-science/>

InvenioRDM/Zenodo, Episciences, Dagstuhl, SwMath, DataCite, Software Heritage and in other infrastructures in the ecosystem, such as HAL and IPOL.

Moving forward, we must continue to develop services, features, and tools to support the different actors, facilitating the adoption of the platforms and making a significant change in academia towards recognizing software source code as a first-class output.

While improving tools, protocols, and workflows, we emphasize the need to support the staff managing the infrastructures. This includes offering training programs, creating detailed documentation, and conducting outreach activities. By equipping support staff with the necessary knowledge and resources, we can improve their effectiveness and ensure the interconnections of the research software ecosystem.

1.1.1 Standardisation and curation definitions

Identifiers and metadata standardisation and curation are essential components of an effective research ecosystem. These processes ensure metadata quality, consistency, and accessibility across organizations and research fields.

Standardisation establishes uniform formats, protocols, and procedures for collecting, storing, and sharing metadata that describes software records. This process creates a common language, making it easier to integrate, compare, and analyze information from various sources. Standardisation in FAIRCORE4EOSC includes, the following activities:

1. Submitting the SWHID specifications: Developing with the [Software Hash ID Working Group](https://www.swhid.org/)⁴ the specifications and proposing the SWHID specifications to a globally recognized organization, such as ISO, to establish a high-quality standard for software identification. This ensures that the standard is developed through a consensus of experts, regulators, and governments, and is effectively implemented in information technology.
2. Developing metadata schemas: Creating standardized structures for metadata using the existing CodeMeta initiative and controlled vocabularies to ensure consistency across platforms. This allows for uniformity in how metadata is recorded and shared, making metadata more reliable and interoperable, while making software more findable and usable.
3. Standardizing API formats: Implementing common API formats for metadata exchange to improve interoperability between systems. This standardisation allows different systems to communicate and share data seamlessly.
4. Mapping metadata standards: Creating mappings between different metadata schemas in the CodeMeta crosswalk table to enable translation between schemas. This ensures that metadata can be easily translated and understood across various platforms and systems.

⁴ <https://www.swhid.org/>

Metadata curation is the active and ongoing management of metadata throughout the software lifecycle. It includes activities aimed at reviewing, enhancing, preserving, and adding value to metadata, ensuring its long-term accessibility and discoverability. In general, curation encompasses, but is not limited to, the following activities:

1. Metadata acquisition: The process of collecting relevant metadata during the creation, submission, or update of research outputs, such as publications, datasets, or software.
2. Metadata review: Checking the acquired metadata to maintain the quality and integrity of metadata.
3. Metadata enrichment: Enhancing metadata by adding additional information, such as context, keywords, or relationships to the reference publication, datasets or software dependencies. This enrichment process adds value to the metadata, making it more useful and easier to discover.
4. Preservation of metadata: Implementing strategies to ensure that metadata is preserved along with the software it describes, ensuring that both remain accessible and usable in the long term. This might include storing metadata in multiple formats or locations to guard against data loss.
5. Community engagement: actively involving the community in the curation process by seeking feedback, encouraging contributions, and fostering collaboration. Engaging with users and stakeholders helps to keep metadata relevant and aligned with the needs of the institution, discipline and the broader research community.

1.2 EOSC Scholarly Infrastructures for Research Software recommendations

In 2020, the Task Force on Scholarly Infrastructures of Research Software, as part of the Architecture Working Group (WG) of the European Open Science Cloud (EOSC) Executive Board, established a set of recommendations in the published SIRS report to allow EOSC to include software, next to other research outputs like publications and data. Four major aspects related to software are analyzed: archiving; referencing (i.e., making identifiable); describing and citing; these are referred to as the ARDC main aspects. The report summarizes the state of the art and identifies best practices for aggregators and catalogs, publishers, and scholarly repositories.

1.2.1 Summary of curation recommendations and gaps

The SIRS Gap Report (Azzouz-Thuderoz et al., 2023) aimed to assess the current state of infrastructures in relation to the recommended requirements from the SIRS report. Appendix A summarizes the relevant recommendations concerning identifiers, metadata curation, and standardisation. Figure 2 illustrates the connected ecosystem as described in the SIRS report.

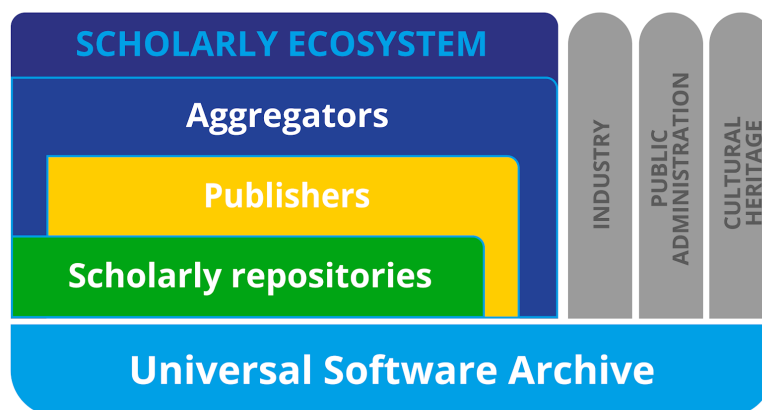


Figure 2 - Architecture of interconnected scholarly infrastructures supporting archival, reference, description and citation of (research) software source code

If the SIRS report emphasizes the need for interoperability between existing infrastructures, it also highlights the need for guidance. In this context, publishers could play a specific role: the four main aspects of software (archiving, referencing, describing, and citing) are not yet fully adopted by researchers, who may not know what is expected of them, as measured in (Garijo et al., 2024). Publishers could take on an educational role, raising awareness about these issues. A concrete step for publishers could be to collaborate with research communities to develop clear guidelines on how to properly cite software (van de Sandt et al., 2019).

The SIRS report also highlights the tight link between citation and metrics and calls for a qualitative approach of evaluating research software. The report emphasizes: “One would need to carefully consider all the implications before promoting purely numeric indicators for software, especially now that a growing international consensus is emerging around principles that value qualitative criteria over quantitative ones, like DORA (Declaration On Research Assessment, 2013) [...]” (EOSC Executive Board & EOSC Secretariat, 2020). A key point is that software may have an important impact outside of academia, where citations don’t count.

1.2.2 Standardisation and curation implementation plan in FAIRCORE4EOSC

The FAIRCORE4EOSC project was launched in June 2022 with the aim to develop core components for EOSC to support a FAIR EOSC and address the gaps identified in the Strategic Research and Innovation

agenda (SRIA). Work Package 6 (WP6), *Services and tools to archive, reference, describe and cite research software*, is implementing strategies to better integrate the existing infrastructures in the scholarly ecosystem to provide services related to Research Software outputs.

Task 6.4 (T6.4), *Metadata and PIDs for Software: Curation and Standardisation*, is part of WP6 in the FAIRCORE4EOSC project. Its primary objective is to develop curation mechanisms to ensure high-quality metadata will be designed and implemented in scholarly repositories and publisher workflows in collaboration with the FAIR-IMPACT project. T6.4 will also contribute to the enhancement of the governance and standardisation efforts related to the adoption of CodeMeta into schema.org and to the ISO standardisation of the SWHID.

1.3 Who is the intended audience of this document?

The complete document, including metadata curation workflows, infrastructure implementation, standardisation efforts, and the standardisation & curation roadmap proposal, is intended for the EOSC community as a project output and for the European Commission as an evaluation of FAIRCORE4EOSC's efforts and results. The roadmap proposal can serve as a strategic action plan for future funding cycles. The intended audience includes institutional policymakers, science clusters, and infrastructure management committees responsible for making decisions on the adoption and implementation of the roadmap.

The content can be also used by the infrastructure team, including the platform/service manager, the community manager and the infrastructure director or management. The information available in this document can assist in the following way:

- Identify existing or potential workflows and tools that can be reused, assess specific workflows and use cases, compile a list of tools with their capabilities, ensure interoperability through shared metadata formats, and provide additional strategies and tips at the service provider level.
- Assess the current infrastructure level of service with Research Software metadata curation and community standards, identify future roadmap directions, provide a high-level overview and outline specific activities to include in the budget and roadmap.
- Build bridges between technical achievements and adopters, highlight the community's role in adopting features and infrastructures, and outline specific activities the community can engage in to better understand the services and features available for researchers.

2 STANDARDISATION

2.1 The CodeMeta metadata journey

2.1.1 What is CodeMeta?

CodeMeta is an initiative designed to improve the way software, particularly research software, is described and cataloged through the creation of semantic artifacts. The initiative has developed key tools including the **CodeMeta vocabulary**, which is a specialized subset of schema.org tailored for software metadata, and the **CodeMeta crosswalk table**, which helps map the broader metadata landscape for better interoperability and integration across different systems. CodeMeta also allows to have a dedicated intrinsic metadata file in the source code itself for descriptive metadata.

Table 1 - Intrinsic vs. Extrinsic metadata

	<i>Intrinsic metadata</i>	<i>Extrinsic metadata</i>
Definition	Metadata about software that is shipped as part of the source code of the software itself or as part of related artifacts (e.g., revisions, releases, etc) is called intrinsic metadata. For example, metadata that is shipped in PKG-INFO files for Python packages, pom.xml for Maven-based Java projects, debian/control for Debian packages, metadata.json for NPM, etc. The citation.CFF and codemeta.json files are also considered intrinsic metadata.	Metadata about software that is not shipped as part of the software source code, but is available instead via out-of-band means. For example, homepage, maintainer contact information, and popularity information ("stars") as listed on GitHub/GitLab repository pages or metadata record in scholarly infrastructure, such as: aggregators, publishers and scholarly repositories.

2.1.2 The CodeMeta Community

The CodeMeta community is central to the development and maintenance of the initiative's resources. This community-driven approach ensures that the tools and standards developed by CodeMeta continue to evolve in response to the needs of the research and software development communities. The main areas of focus for the community include:

- **The CodeMeta Vocabulary:** This vocabulary serves as the core framework for describing software metadata. The community actively maintains and updates this vocabulary to ensure it stays current with the latest needs and trends in software description.
 - Repository: [CodeMeta Vocabulary](#)
- **The CodeMeta Crosswalk Table:** This table maps metadata from various sources to the CodeMeta vocabulary, facilitating interoperability between different metadata standards.

- Repository: [CodeMeta Crosswalk Table](#)
- **The CodeMeta Website:** This site provides comprehensive resources and documentation for users and contributors, including access to the vocabulary, crosswalks, and other tools.
 - Repository: [CodeMeta Website](#)
- **The CodeMeta Generator:** A tool that helps users generate CodeMeta metadata for their software projects, making it easier to create standardized descriptions.
 - Repository: [CodeMeta Generator](#)

2.1.3 A collaborative governance model

The CodeMeta governance implementation involved several key steps. First, the draft governance policy was reviewed, introducing a Project Management Committee (PMC) and Committers to replace the role of maintainers, and establishing a lightweight, community-driven decision-making process. Next, key organizers, including Matt Jones, Arfon Smith, Abby Cabunoc Mayes, and Carl Boettiger, accepted the governance proposal⁵. The PMC was then formed, with current organizers given the option to stay and additional active community members invited to join. Following the CodeMeta PMC [vote in May 2023](#), the current chair of the PMC is Morane Gruenpeter.

A list of proposed committers was created, shared with the PMC for feedback. A mailing list was set up for the PMC, with community members encouraged to open tickets for discussions, while external mailing lists like the Force11 Software Citation Implementation WG⁶ and the SciCodes consortium⁷ were identified for promotion. Existing PRs were reviewed and categorized for further discussion, approval, or direct merging based on their impact on the specification. A governance repository was created under the CodeMeta organization, and the main repository was updated with roles and governance information. Plans were made to automate website updates and inform the community about the new governance policy.

Community voting on issues and PRs was scheduled, with key dates set for review and voting. PRs were then to be merged and published as part of the v2.1 and v3 releases. Finally, if additional discussions were needed based on voting outcomes, an online workshop was planned.

The community's efforts are documented and shared openly, allowing for continuous feedback and improvement. This governance model fosters a collaborative environment where the community can contribute to the ongoing improvement of software metadata standards, ensuring that CodeMeta remains a valuable resource for the global research community.

⁵ <https://codemeta.github.io/governance/>

⁶ <https://force11.org/groups/software-citation-implementation-working-group/>

⁷ <https://scicodes.net/>

2.1.4 From CodeMeta v2.0 to CodeMeta v3.0

The CodeMeta description format is continuously evolving to align with the evolving requirements of the research software ecosystem and the needs of users within scholarly infrastructures. As part of FAIRCORE4EOSC's efforts, following the establishment of the governance model, a key focus was on coordinating the community to finalize version 3.0 of the CodeMeta vocabulary.

In 2023, version 3.0 has been published, adding the following new vocabulary elements:

- review, which allows you to give [review information](#) about the software, in this case reviewAspect and reviewBody.
- role, which, associated with an author or a contributor, allows you to define the function (roleName) that this person has held, and for what period of time (startDate and endDate).
- hasSourceCode adding a link that states where the software code is for a given software.
- isSourceCodeOf adding a link that states where a software application is built from a given source code. This is the reverse property of hasSourceCode.

In v3.0, some properties also changed name for clarification:

- *contIntegration* became *continuousIntegration*
- *embargoDate* became *embargoEndDate*

Just as translation files exist for converting between various metadata description formats, there is also a file⁸ for converting from format v2.0 to v3.0.

2.2 Identifying source code: The Software Hash Identifier

Software is particularly difficult to identify in the long run: there can be multiple versions over the years, but also, names may change, and projects can migrate from one platform to another during the development process. Software may be a resource difficult to identify also because of its complex architecture. Even the most simple software is in fact based upon multiple sub-components and in some cases, one needs to reference a specific sub-item rather than the whole software. In some cases, it may be accurate to reference a step of the development process itself, such as a commit.

The DOI is a PID widely used by researchers to identify academic outputs, including data sets. But software calls for specific solutions. As highlighted by Di Cosmo and his co-authors, “[...] we need identifiers that are not only unique and persistent, but also support integrity in an intrinsic way” (Di Cosmo

⁸ <https://github.com/codemeta/codemeta/blob/master/crosswalks/codemeta-V2.csv>

et al., 2018): in other terms, such PIDs don't rely on any central registry nor naming authority. The only trusted component is the algorithm that is used to compute the identifier from the object itself. SWHIDs can be computed using cryptographically strong functions directly from the digital objects they refer to, by anyone that has access to a copy of those objects.

The SoftWare Hash Identifier (SWHID) is an intrinsic identifier designed for software source code archived in Software Heritage. Currently, over 30 billion software artifacts in the Software Heritage archive are identified using SWHIDs. SWHIDs are based on a cryptographic hash, which is used in a Merkle tree structure, providing a specific identification for wide range of software artifacts⁹.

To understand the relationship between intrinsic identifiers (SWHIDs) and extrinsic identifiers (HAL-ID, DOI, etc.) we need to separate referencing and citation, as detailed in the table below from the moderation guide to software deposits (Granger et al., 2022) and the FAIR-IMPACT D4.4 (Gruenpeter et al., 2023):

Table 2 - Intrinsic vs. Extrinsic identifiers (extended in [Appendix D](#))

	Intrinsic identifiers (SWHID)	Extrinsic identifiers (DOI, HAL-ID, etc)
Objective	Reference	Cite
	The two objectives are related but not dependent on each other. For example, to ensure the reproducibility of research, referencing the software artifacts is sufficient.	
Use case	<p>To capture the digital footprint of artifacts resulting from software development (e.g., source code, executable binaries, etc.) in order to ensure their reuse. The identification process should address different levels of granularity. We describe digital objects.</p> <p>Retrieve byte-identical copies of source code artifacts.</p>	<p>To identify the authors of software and attribute the credit of its creation to these authors. We describe the project that resulted in digital objects, but the project itself is not a digital object. We attribute the credit of a work to its authors: this is why a manual verification step is necessary</p> <p>Refer to a given software in a catalog, a citation Track the software produced by an institution.</p>

⁹

<https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html#choosing-what-type-of-swhid-to-use>

2.2.1 SWHID Working group and governance

SWHIDs can be resolved using the Software Heritage Web interface but they can be used independently from Software Heritage. SWHIDs don't depend on a given version control system. Added to this, SWHIDs were designed for source code but are not restricted to it: SWHID goes way beyond source code and Software Heritage. SWHIDs are for instance used in [the SPDX standard from version 2.2](#).

Considering the broad range of applications of the SWHIDs, an open process has been put in place in order to produce a publicly available specification that describes precisely how these identifiers are defined and computed, easing adoption for all stakeholders, and assembling a working group to produce an open specification. The kick-off meeting took place on March 27th 2023 (Di Cosmo, 2023). Further information is available on the SWHID working group website: <https://swhid.org/>

The governance policy for the SWHID Project is derived from the [Community Specifications Governance Policy 1.0](#). Participation in the elaboration of the SWHID standard is open to all. The project's work is divided up into multiple separate Git repositories, which can be found at <https://github.com/swhid>. The project makes decisions through a consensus process ("Approval" or "Approved"). While the agreement of all Participants is preferred, it is not required for consensus.

Table 3 - SWHID Working Group roles

Role	Description
Participants	They contribute to the project via issues and pull requests on GitHub.
Core team ¹⁰	The body that is responsible for governing the SWHID Project: coordinates ; makes decisions when community consensus cannot be reached ; adopts and maintains policies or rules and procedures for the Project as appropriate.
Maintainers	Actors in charge of organizing activities around developing, maintaining, and updating the specification(s) developed by the Working Group. They are designated by the Core team. One of the tasks of the maintainers is to train new participants.
Editors	Actors in charge of ensuring that the contents of specifications produced by the Working Group are complete, coherent, and adhere to applicable formatting and content guidelines.

¹⁰ Current core team composition: Jean-François Abramatic, Roberto Di Cosmo, Morane Gruenpeter, Stefano Zacchiroli, Alexios Zavras.

2.2.2 Specifications and ISO submission

The Software Hash Identifier specification defines an open standard for content-based identifiers of digital objects. The specification aims at removing ambiguities so that any user with the accurate skills may implement the same calculation algorithm. This standard defines the syntax, description and precise computation process for a SWHID, which is intended to provide unique identification, guarantee integrity and enable provenance tracking of digital objects that may have, and be part of, a hierarchical structure. The first stable version of the specification is publicly available: <https://www.swhid.org/specification/>.

The specification development process follows 4 stages :

Table 4 - SWHID specifications development process

1	Draft	Each updated version of the SWHID Specification following the merging of a pull request will be considered a "Draft Specification"
2	Approved Specification status	The core team considers a draft version to become a candidate for "Approved Specification" status. After the review and upon the Approval of the Working Group, the document status is updated.
3	Publication	The document and the terms under which the Approved Specification is being made available are publicly shared.
4	Submission to standard bodies	Only Approved Specifications may be submitted to another standards development organization.

The SWHID specification was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see <https://www.iso.org/directives>). ISO (the International Organization for standardisation) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. The specification is currently in review of the ISO/IEC JTC 1 technical committee, ISO/IEC DIS 18670: SoftWare Hash IDentifier (SWHID) Specification V1.2¹¹.

¹¹ <https://www.iso.org/cms/%20render/live/en/sites/isoorg/contents/data/standard/08/99/89985.html>

2.2.3 SWHID adopters

- the [Software Package Data Exchange](#) (SPDX) specification includes SWHIDs in its published [version 2.2](#)
- the [swh_prefix used in SWHIDs is registered](#) with [IANA](#)
- The WikiData property [P6138](#)
- Scholarly repositories
 - HAL
 - Zenodo (in a few weeks in production)
- Publishers
 - [Image Processing On Line journal](#) (IPOL) provides [reference to software code](#)
 - [eLife](#), an open-access journal [saves code in SWH](#)
 - Episciences, the the CCSD publication platform provides links to software [on article submission using SWHID](#)
 - Dagstuhl (example on supplementary materials section in [DROPS](#) article)
- Aggregators
 - SwMath, the mathematical registry [collaborates with SWH for referencing software](#)
 - OpenAire
- Other infrastructures
 - Replicability Stamp on the [Computer Graphic Replicability Stamp Initiative \(GRSI\)](#) platform has adopted Software Heritage and the SWHIDs
 - GNU Guix is a package manager for GNU/Linux systems that allows users to control and reproduce computing environments easily across multiple devices. GUIX has [adopted SWHIDs since 2018](#).

2.2.4 SWHID tools

Software Heritage resolver

SWHIDs can be resolved using the Software Heritage [Web interface](#). In particular, the root endpoint / can be given a SWHID and will lead to the browsing page of the corresponding object, like this:

- <https://archive.softwareheritage.org/<identifier>>.

A dedicated /resolve endpoint of the Software Heritage [Web API](#) is also available to programmatically resolve SWHIDs; see:

- [GET /api/1/resolve/\(swhid\)/](#).

Third-party resolvers

The following third party resolvers support SWHID resolution:

- [Identifiers.org](#); see: <http://identifiers.org/swh/> (registry identifier [MIR:00000655](#)).
- [Name-to-Thing \(N2T\)](#)

Locally resolving a SWHID

To compute a Software Hash ID (SWHID) locally, follow these steps:

First, install the necessary tools by running `pip install swh-model[cli]`.

Then, use the `swh identify` command to generate SWHIDs for files or directories.

- For example, running `swh identify fork.c kmod.c sched/deadline.c` will compute unique SWHIDs for each file,
- while `swh identify --no-filename /usr/src/linux/kernel/` will generate an SWHID for the entire directory.

Additionally, you can identify repositories using `swh identify --type snapshot` after cloning them with Git. Ensure that the software is archived in Software Heritage if you want others to resolve the SWHIDs.

Warning: If you expect others to be able to resolve the SWHIDs of source code you care about, you should make sure the corresponding software is archived in Software Heritage.

For more information on the command-line, see the [Software Heritage documentation](#).¹²

¹² <https://docs.softwareheritage.org/devel/swh-model/cli.html>

2.3 Community

The standardisation of research software metadata is not just a technical challenge—it's a community-driven effort. The success of initiatives like FAIR-IMPACT and FAIRCORE4EOSC hinges on the active participation and collaboration of diverse stakeholders across the research ecosystem. In this section, we delve into how community-driven projects and task forces, particularly those within the EOSC framework, are shaping the future of research software management.

2.3.1 EOSC projects & task forces

Community efforts are the backbone of any meaningful standardisation process. The work done by EOSC projects and task forces, such as the development of the Research Software MetaData Guidelines in FAIR-IMPACT project or the quality-driven initiatives led by the Infrastructures for Quality Research Software Task Force¹³, exemplifies the power of collective action. These initiatives don't just create standards—they build the infrastructure and cultural frameworks necessary to embed these standards into the daily practices of researchers.

FAIR-IMPACT

The RSMD (Research Software Metadata) Guidelines, developed by Task 4.3 within the FAIR-IMPACT project, provide a comprehensive and adaptable framework for managing research software metadata. These guidelines are designed to be flexible, catering to end-users across various disciplines and software development contexts. The deliverable includes several key components: an introduction of goals, methodology, and use cases; a state-of-the-art review of existing practices and guidelines in the field; a comprehensive analysis of the current metadata landscape; a proposal for RSMD guidelines that offer structured recommendations for collecting and curating research software metadata; and a clear checklist for researchers to ensure compliance with best practices in metadata management. Additionally, a living version of the RSMD Guidelines will be maintained in a dedicated repository¹⁴, allowing for ongoing updates and improvements as the field evolves. These guidelines serve as a valuable resource for researchers and institutions aiming to enhance the findability, accessibility, interoperability, and reusability (FAIR) of research software through effective metadata management.

By ensuring that research software is properly described with standardised metadata, the RSMD Guidelines contribute to the creation of a more interoperable and accessible research infrastructure, which is a central aim of FAIRCORE4EOSC. This connection underscores the importance of metadata curation as a foundational element in achieving the FAIR principles within the research software community.

¹³ <https://eosc.eu/advisory-groups/infrastructures-quality-research-software/>

¹⁴ <https://github.com/FAIR-IMPACT/RSMD-guidelines>

EVERSE

[EVERSE](https://everse.software/)¹⁵ aims to build a European network for Research Software Quality and establish a foundation for a future Virtual Institute for Research Software Excellence. The EVERSE project and FAIRCORE4EOSC are working together to improve research software quality and infrastructure within EOSC.

The FAIRCORE4EOSC-EVERSE collaboration is focused on aligning and adopting tools and services for archiving, referencing, describing, and citing research software. The partnership follows recommendations from the EOSC SIRS report to connect scholarly repositories with the Software Heritage archive, using standards like CodeMeta and Software Heritage identifiers (SWHIDs). Planned activities include EVERSE reviewing key deliverables from FAIRCORE4EOSC, sharing access to BETA versions of FAIRCORE4EOSC components, and hosting joint events like workshops and webinars. This collaboration aims to support the broader goals of Open Science and strengthen the EOSC ecosystem.

Infrastructures for Quality Research Software

The Infrastructures for Quality Research Software Task Force¹⁶, co-chaired by Roberto Di Cosmo (INRIA) and Isabel Campos (CSIC), was focused on advancing the development and deployment of tools and services that enable researchers to effectively archive, reference, describe, share, and reuse research software. The Task Force aimed to improve software quality from both technical and organizational perspectives. By engaging with scholarly infrastructure providers, particularly those involved in EOSC-related projects, the Task Force explored tools, standards, and platforms used in software development and quality control, and formulated actionable recommendations.

Key documents published during 2023 and 2024 by the Task Force included,

- "Ensure Software Quality" (DOI: 10.5281/zenodo.10723608),
- "Review of Software Quality Attributes and Characteristics" (DOI: 10.5281/zenodo.10647227),
- "SIRS Gap Analysis Report" (DOI: 10.5281/zenodo.10376006),
- "Research Software Lifecycle" (DOI: 10.5281/zenodo.8324828),
- "Review of Software Quality Attributes and Characteristics" (DOI: 10.5281/zenodo.8221384).

These documents reflected the Task Force's efforts to identify best practices, ensure quality, and provide both qualitative and quantitative methodologies for unbiased quality assessment in research software.

¹⁵ <https://everse.software/>

¹⁶ <https://eosc.eu/advisory-groups/infrastructures-quality-research-software/>

2.3.2 International community driven initiatives

SciCodes consortium

The SciCodes initiative was launched in 2021 to develop a shared approach to software inventory. The SciCodes Consortium welcomes publishers and managers of catalogs and institutional research software infrastructures. The objectives of SciCodes are:

- Share work methods, marketing ideas and communication practices,
- Present specific aspects of the respective services of its members, discuss challenges and share solutions to common problems that arise in resource management,
- Work collaboratively to accelerate adoption of CodeMeta and CFF standards and enable better software citation, recognition and dissemination,
- Work towards a universal standard to enable search across multiple software registries and interoperability across various infrastructures and platforms.

The consortium was initially established by members of the “Best Practices for Software Registries” task force in January 2021, formed from the FORCE11 SCIWG working group.

The task force organized a workshop in 2019, the discussions of which were synthesized and published in the report by the Task Force on Best Practices for Software Registries et al. (2020). The consortium had an international reach, with significant infrastructure representation from the United States.

As part of these best practices, the following steps are recommended:

- Provide a public scope statement
- Provide user guidance
- Provide guidance to software contributors
- Establish an authorship policy
- Share metadata schema
- Stipulate conditions of use
- State a privacy policy
- Provide a retention policy
- Disclose your end-of-life policy

The consortium holds meetings on the 3rd Thursday of each month, which are open to all. A full list of members is available online¹⁷.

¹⁷ <https://scicodes.net/participants/>

RDA

The Research Data Alliance (RDA) Software Source Code Interest Group provides a forum to discuss issues related to the management, sharing, discovery, archiving, and provenance of software source code, with a particular focus on source code that generates research data and plays a crucial role in scientific publications. Founded in 2017, following Plenary P9 in Barcelona Birds of Feather meeting¹⁸.

In May 2023, during Virtual Plenary VP22, discussions focused on strategies to increase the adoption of scholarly infrastructures for software. Ideas from these discussions included:

- Providing clear illustrations of the benefits of adopting software archival practices to encourage their use.
- Requiring software management plans (in addition to data management plans) for research projects.
- Offering education options, particularly for undergraduate and graduate students, by providing training for using software archival infrastructures early in their careers.
- Announcing plans to revisit the FAIR for Research Software Working Group (FAIR4RS WG) 2022 outputs.

Collaborations with the larger community are ongoing, especially with the SciCodes consortium, which focuses on repositories and registries that support software outputs in the scholarly ecosystem.

ReSA

The Research Software Alliance (ReSA), as presented in (Katz & Barker, 2023), envisions a world where research software and those who develop and maintain it are recognized and valued as fundamental to global research. ReSA's mission is to advance the research software ecosystem by collaborating with decision-makers and key influencers. Through its Task Forces and Funder Forum, ReSA actively engages with stakeholders to drive cultural change, advocate for the funding of critical infrastructures, and promote standards that support reliable, reproducible, and sustainable research software. ReSA is a fiscally sponsored project of Code for Science & Society and is led by the ReSA Steering Committee.

The importance of these community efforts cannot be overstated. They are essential in creating a research environment where software is recognized as a first-class output, complete with the robust metadata, quality control, and interoperability that the modern research landscape demands. Without the engagement and commitment of the community, these standards would remain theoretical exercises rather than practical tools for improving research outcomes.

¹⁸

<https://web.archive.org/web/20240224222930/https://www.rd-alliance.org/software-source-code-focus-group-rda-9th-plenary-bof-meeting#expand>

3 CURATION WORKFLOWS

3.1 State of the art of curation workflows

Software can be developed on code hosting platforms, also known as forges, which allow the code to be publicly available from the inception of the research output or at any point during the development timeline (Le Berre et al., 2023). However, forges are not designed for long-term sustainability and are not intended to serve as archives, as we saw with GoogleCode¹⁹, CodePlex²⁰, Gitorious²¹ and other commercial and institutional examples.

To ensure long-term access to research software, it should be preserved in an archive, such as Software Heritage, or/and deposited in institutional repositories, as recommended in (Directorate-General for Research and Innovation (European Commission) & EOSC Executive Board, 2022; EOSC Executive Board & EOSC Secretariat, 2020). These scholarly repositories, such as HAL and Zenodo, provide a long-term, stable location for storing and sharing research software along with their respective metadata records. This is particularly important for supporting the reproducibility of research results, as it guarantees access to the software used in a particular study even if the original code is no longer available elsewhere.

While some scholarly repositories offer long-term archival services, it's important to recognize that software curation involves specific workflows with varying modalities. These modalities differ in terms of the expected level of curation, archival quality, and the availability of persistent identifiers (PIDs), as outlined in Table 5. Additionally, there is a significant difference in the dissemination process depending on whether the software is simply made public or formally published.

Curation quality, in this context, specifically refers to the accuracy and completeness of the curated metadata. In addition, Table 5 outlines other levels of curation that include peer review.

Table 5 - Dissemination modalities in metadata curation workflows

Dissemination modalities	Curation level	Archival level	PIDs available
Not public	Depend on institution	Depend on institution	-
Public on forge	No curation required by the infrastructure	No archival	No PID

¹⁹ <https://www.theverge.com/2015/3/13/8206903/google-code-is-closing-down-github-bitbucket>

²⁰ <https://devblogs.microsoft.com/bharry/shutting-down-codeplex/>

²¹ <https://www.softwareheritage.org/2016/07/21/gitorious-retrieved/>

Public and available on the Software Heritage (SWH) archive	No curation required by the infrastructure	All dev history archived	Intrinsic PID - the SWHID
Public and self archived in institutional repository (without curation)	No curation	Deposited version archived in institutional repository	Extrinsic PID on institutional repository (e.g DOI)
Public and submitted in institutional repository with curation (as .zip / .tar.gz deposit)	Verification of metadata by moderator (the HAL example)	Long term archiving of the deposited version	Extrinsic PID to access metadata record and direct reference to the source code with intrinsic PID (SWHID)
Public and submitted in institutional repository with curation (using the SWHID)	Verification of metadata by moderator (the HAL example)	All dev history archived in SWH and reference added to the metadata record	Extrinsic PID to access metadata record and direct reference to the source code with intrinsic PID (SWHID)
Published - submitted to publisher platform + peer review (minimal)	Metadata curation without code review of the source code, only review of the associated article (Episciences example)	The option of software archiving is offered at SWH	Receive PID upon publication.
Published - submitted to publisher platform + peer review (medium)	Metadata curation with medium review of the software - can be installed and executed		
Published - submitted to publisher platform + peer review (good)	Metadata curation with good review of the software - submission of software, developed and		

	documented properly, works as expected		
Published - submitted to publisher platform + peer review (best)	Metadata curation with best review of the software - an integral part of the scientific publication ensuring research quality with enhanced review of the source code (Graphics Replicability Stamp Initiative (GRSI) example)	The option of software archiving is offered at SWH	PIDs are assigned to the publication of the software description.

The peer review evaluation used in the table (minimal, medium, good and best) are proposed in the SIRS report as part of the Credit recommendations emphasizing the importance of peer review for research evaluation.

3.1.1 Sharing and self-archival

Self-archival is a process where researchers independently deposit their research software into an archive or repository without the involvement of external moderation or curation. This approach allows researchers to safeguard their software and ensure it is citable and accessible to the broader community. While self-archival offers the advantage of speed and autonomy, it also places the responsibility on the researcher to ensure that the metadata is accurate and that the software is properly documented. This method is particularly useful for researchers who wish to rapidly disseminate their work, although it may lack the additional layers of curation quality provided by curated or moderated archival processes. In the context of long-term preservation and accessibility, self-archival can be a valuable tool, especially when combined with repositories that offer robust support for metadata standards and persistent identifiers (PIDs).

3.1.2 Dissemination and metadata curation or moderation: the HAL story

The collaboration between Software Heritage (SWH), Inria and the CCSD enabled the opening of the French open archive HAL toward the new type of scientific data: the Software, as described in (Di Cosmo,

Gruenpeter et al., 2020). Researchers now have the opportunity to deposit the source code via HAL and contribute to the Software Heritage archive.

Depositing research software in HAL with minimal curation of the metadata involves submitting the software as a compressed file (e.g., .zip or .tar.gz), where it undergoes a review process to ensure quality and completeness of the metadata in comparison to the deposited artifact. This curation step verifies that the software and its metadata meet certain standards before being made publicly accessible. Unlike self-archival, this method offers enhanced credibility, better discoverability, and ensures that the software is properly preserved for long-term access. It also often includes the assignment of a persistent identifier (PID), making the software easier to cite in academic publications. This approach provides a balance between rapid dissemination and the assurance of quality and longevity.

The deposit workflow on the HAL platform and archiving into SWH

The complementary strengths of Software Heritage and HAL provide several key benefits:

- **Archiving:** Software Heritage offers long-term archiving, ensuring that software remains accessible on a dedicated platform, which supports the reproducibility of scientific work and preserves current scientific knowledge for future generations.
- **Identification:** The SWHID (Software Heritage ID) allows precise referencing of the archived software on Software Heritage, facilitating reproducibility by linking to the exact content, while the HAL repository provides an additional identifier with descriptive metadata.
- **Description:** Detailed and verified metadata enhance the visibility of software by providing specific descriptions that meet quality standards.
- **Citation:** Comprehensive citation information, available in multiple export formats, supports interoperability and strengthens links between publications, data, and source code.
- **Promotion:** HAL offers additional services for promoting software, including various export formats for integration into bibliographies, activity reports, and websites, enhancing the dissemination of scientific output.
- **Credit:** The system allows proper credit to be given to authors according to their specific roles in software development.

In HAL, the expected submissions are source code files under a free license, excluding executables, and deposits are reserved for the academic community. Only the authors or creators of the software and their representatives are permitted to deposit on HAL.

There are two ways to register software with HAL:

Tarball deposit

The files are compressed (.zip, .tar.gz) and addition of metadata (authors and affiliations, license, etc.). Once posted on HAL, the source code is transferred to Software Heritage which assigns a SWHID. The HAL repository is then updated with the SWHID.

SWHID deposit

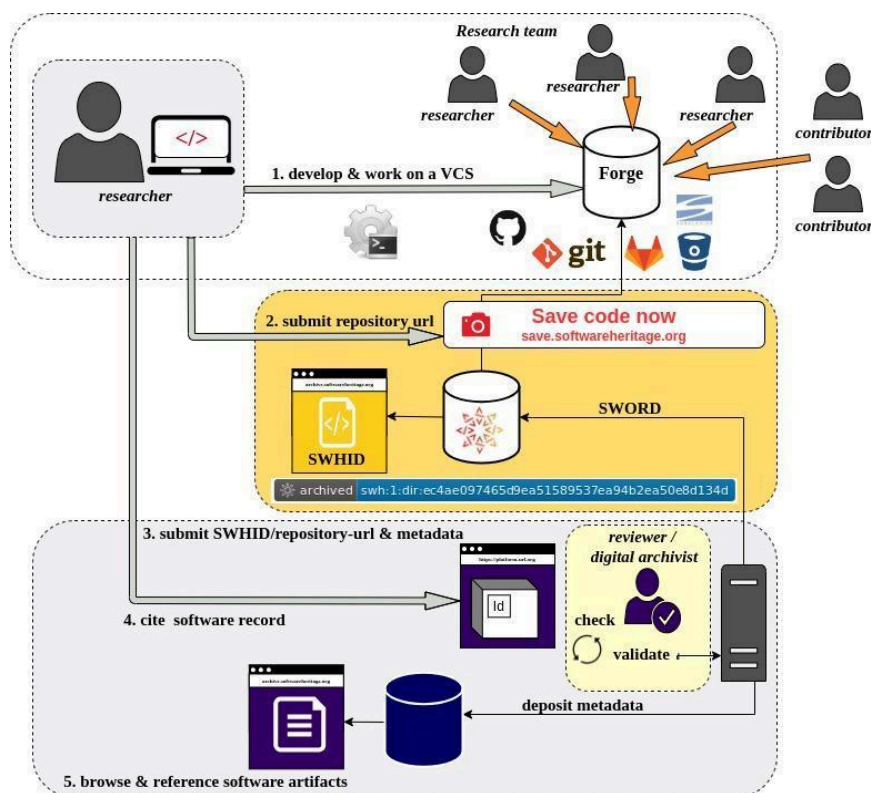


Figure 3 - Deposit workflow using a SWHID on scholarly infrastructure

The SWHID deposit workflow is applicable when the source code has already been archived in Software Heritage, as shown in Figure 3. In submitting to a scholarly repository like HAL, this identifier is used along with completing the associated metadata. If a codemeta.json file is available in the code, HAL will automatically extract the metadata to facilitate the deposit process. This automatic process is presented

in Figure 4. The metadata retrieved in HAL includes: Authors, Title, Description, Production/Writing Date, Programming Language, Keywords, Development Status, Version, Code Repository, Licenses, and Identifiers. Ongoing efforts aim to improve metadata extraction by incorporating updates from the latest Codemeta version.

In both cases, the consistency between the metadata and the content of the files is checked before posting online by HAL's moderators.

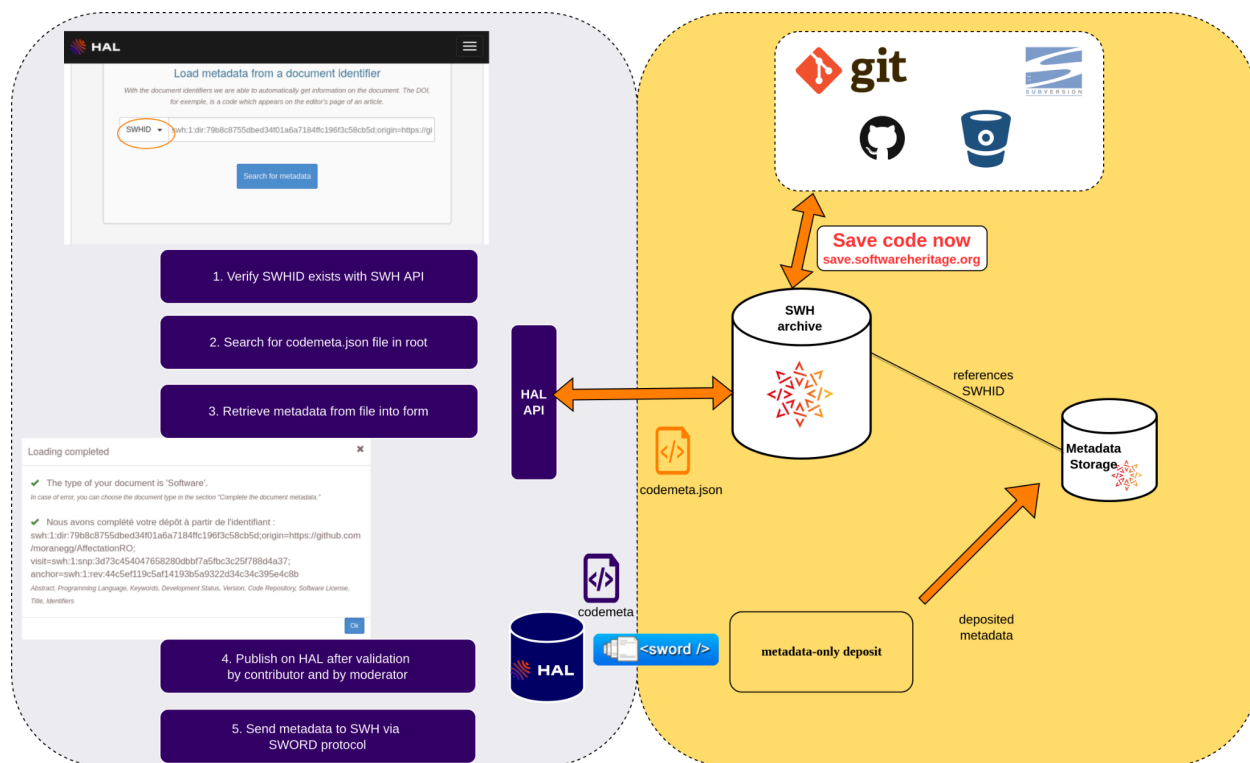


Figure 4 - Extraction automation of metadata using a SWHID on HAL

3.1.3 Publishing and peer review

Publishing software can be considered an academic publication, and submitting the software artifact to an academic publication that has been qualified through some form of peer review can be considered a research result. There are several examples of academic publications that publish research software, including AEC, IPOL, Dagstuhl Artifacts series (DARTS), and the JOSS (Journal of Open Source Software).

The peer review, as described in (EOSC Executive Board & EOSC Secretariat, 2020) has four main levels for research software:

- The best level ensures high-quality research software for scientific publications, such as [DARTS](#)²², [ACM Badging schema](#)²³, and [Image Processing Online \(IPOL\) Journal](#)²⁴ practices.
- The good level checks novelty, development, documentation, and functionality, as seen in the [Journal of Open Source Software \(JOSS\)](#)²⁵.
- The medium level focuses on installation and result reproduction, demonstrated by the [Reproducibility Label](#)²⁶ and Codecheck (Nüst & Eglen, 2021).
- The minimal level ensures proper archiving and referencing without reviewing the source code.

Peer review is essential to ensure the quality and credibility of research software. It involves experts ("peers") evaluating research data and artifacts in the same field. In software development there are other mechanisms to review code, with merge requests and pull request available on distributed version control systems²⁷, these mechanisms are out of scope for this deliverable.

The [Image Processing Online \(IPOL\) Journal](#) example in figure 5, shows the software artifact on the metadata record of the published article. The software was reviewed and tested when the article was submitted.

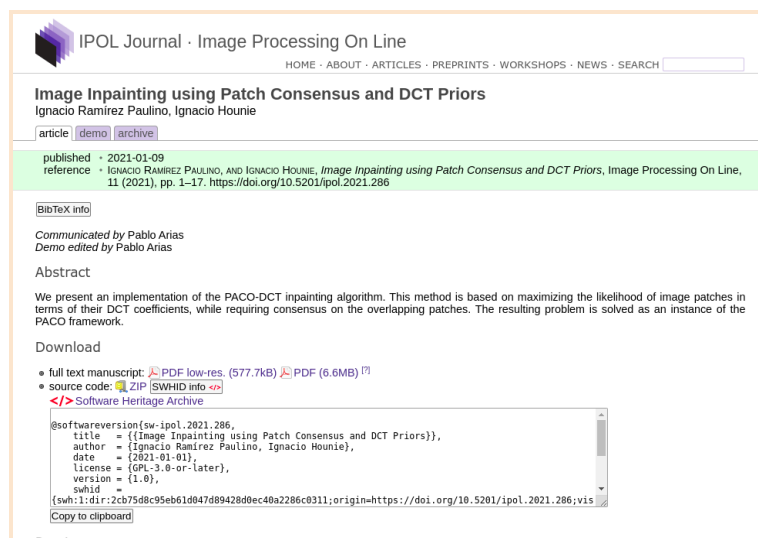


Figure 5 - IPOL journal metadata record - published after peer review

²² <https://drops.dagstuhl.de/entities/journal/DARTS>

²³ <https://www.acm.org/publications/policies/artifact-review-and-badging-current>

²⁴ <https://www.ipol.im/>

²⁵ <https://joss.theoj.org/>

²⁶ <https://rlpr.github.io/>

²⁷ https://en.wikipedia.org/wiki/Distributed_version_control#Pull_requests

Artifact Evaluation as part of the publication process

The Artifact Evaluation (AE) process takes place after the paper decisions have been made in different journals and proceedings, see recent example at [EuroSys 2024](#)²⁸. Authors submit their software artifacts for evaluation by the Artifact Evaluation Committee (AEC). The AEC ensures that software artifacts are well documented, reusable, consistent, and complete as claimed in the associated research paper. There are different badges available after the evaluation, as proposed in the [ACM review and badging](#)²⁹. Software artifacts should include a document explaining their use, reproducibility of experiments with specific references to the paper, system requirements, installation instructions, the software itself, and any related data.

3.2 Curation quality – keeping humans in the loop

Intrinsic metadata can be automatically extracted, as shown in figure 4, to generate descriptions and this workflow may be deemed the most cost-effective solution as no manual operation is required. But to effectively manage research software and give proper credit to its authors, a manual moderation is required, as recommended by the SIRS report (EOSC Executive Board & EOSC Secretariat, 2020). For instance, as the types of the contributions may evolve during the project development, the list of the authors has to be carefully checked.

Activities like reviewing source code functionality, compiling and running the software, or assessing reproducibility and accuracy are outside the moderator's scope. These tasks are typically handled by software developers or specialized experts. **Curation quality is understood here as the curation of the metadata quality.** Plagiarism detection is also a distinct concern, and the SIRS report recommends that publishers have to support standard manual checks as a minimum requirement.

3.2.1 Good enough metadata curation: sharing the cost of the metadata management

In order to support the adoption of good practices related to software, the stake of metadata curation also needs to be considered from an organizational perspective: how to deal with the tension between long-term goals and current practices to determine a cost model? As stated by Treloar and Wilkinson (Treloar & Wilkinson, 2008) about metadata creation in a data-driven research world, comprehensive metadata capture relies on the effort of many stakeholders: output creators, repository managers, and moderators. But defining a too minimalistic set of expectations for descriptions loads the costs on data discoverers. How to find the right balance?

²⁸ <https://sysartifacts.github.io/eurosys2024/index>

²⁹ <https://www.acm.org/publications/policies/artifact-review-and-badging-current>

Capture as much as you can afford: defining a minimum metadata quality level

The case of the French repository HAL provides an example of appropriate sharing of effort between research software authors, moderators, and end-users. The figure illustrates in particular the key actions performed by the metadata moderator. This includes detecting extraneous content, checking consistency between metadata and source code, and completing or correcting deposit metadata. Metadata is first entered by the software author and then checked and enriched by the moderator.

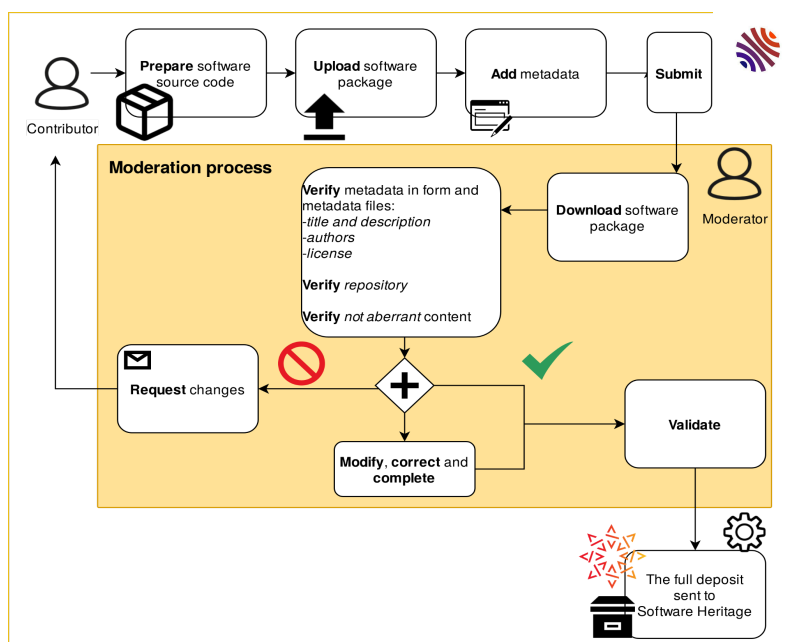


Figure 6 - Figure 6: Curation workflow on HAL

To properly describe the output, software-specific metadata fields had been added to the HAL form³⁰ but as the table below shows, the set of mandatory metadata to complete a deposit is kept minimalistic. The table focuses on tasks performed by the moderator. Note that a deposit may take only a couple of minutes in the case of a depositor who added a CodeMeta file to the code repository and who uses the SWHID deposit.

³⁰ SWHID; programming language; code repository; platform/OS; version ; development status; runtime platform

Table 6 - fields in software records on HAL

HAL mandatory field for software type: i.e. the deposit is not completed if one of the information below is missing	Alternative label (i.e. how the same information may be named in intrinsic metadata)	How to check if the field had been properly filled out: sources of information for the moderator	Comments
Document type	Not applicable	HAL drop down menu	Each document type corresponds to a form template, containing the fields specific to the document to be described.
Software name	Title	README file Code Repository URL CodeMeta file	The moderator checks the consistency of the information among the different sources of information. Inconsistent naming may occur: e.g. in one file, the authors recommend adding a subtitle to the software name when citing. But in the code repository and other information sources, the short version of the name seems to be the main version.
Domain Academic field	Not applicable	HAL drop down menu	The use of a pre-selected list alleviates the work of the depositor and avoids multiplying namings. Added to this, thanks to this common naming, software collocates with different materials types in a results page.
Licenses	COPYING COPYRIGHT	LICENSE or LICENSES files COPYING file COPYRIGHT file Is it an open license? Then, the depositor uses a drop down menu or free text field. In case of multiple licenses, the depositor mentions all of them.	Whatever the deposit process chosen (i.e. SWHID deposit or manual deposit), the moderator scrutinizes any mention of licenses, even if no dedicated files have been provided. In case of multiple licenses, the software author is accountable for their compliance, not the moderator. The moderator's role is to check the consistency between the metadata entered in the form and the sources of information provided.
Name (authors)	CONTRIBUTORS	AUTHORS file	If any inconsistency is spotted, the

HAL mandatory field for software type: i.e. the deposit is not completed if one of the information below is missing	Alternative label (i.e. how the same information may be named in intrinsic metadata)	How to check if the field had been properly filled out: sources of information for the moderator	Comments
		CodeMeta file README file CONTRIBUTORS file CREDITS file CITATION file Code repository citation.CFF	moderator has to check with the depositor if it's done by purpose or not. For instance, one may choose to mention a team name rather than a detailed list of authors. The discussion may be a good opportunity for the moderator to check that the depositor understood all the impacts of such a decision. If the role of the contributor is specified in this file, the moderator can add it in the HAL metadata.

Identifiers are automatically attributed: an HAL-ID is associated with the new resource, once released. In case of a tarball deposit, a SWHID is generated once the export of the metadata and the files to Software Heritage is completed. And in case of SWHID deposit, the identifier is integrated, alongside with the other metadata.

The role of the moderator also involves ensuring the quality and accuracy of metadata associated with software links. This includes verifying the validity of links, such as SWHIDs and Git URLs, and confirming that the content is accessible and complies with open access standards. Moderators help maintain the integrity and reliability of these links, ensuring they are current, functional, and properly documented, which is crucial for effective software citation and reuse in the research community. These actions can and should be automated as much as possible to reduce the effort and manual verification of metadata properties.

Capture as much as you can afford ... but do it with appropriate tools

As stated by Rios (Rios, 2018), service providers such as libraries face a challenging equation: incorporating software into existing data-oriented service workflows while minimizing additional resources. It might be tempting to aggregate data and code together. But treating software as a sub-component of data or as a similar output can't lead to generating accurate descriptions. Therefore, the HAL repository had been adapted to enable proper software description.

Yet, adapting existing tools may still be considered as a resource consuming option. But guidelines without proper workflows that support them have a very limited impact, as pointed out by Mayernik (Mayernik, 2016): "Without routines for data collection, management, and preservation that incorporate

the standard, the standard is little more than a curiosity that exists independently from the day-to-day data management and curation efforts.” The need for tools that automate the application of good practices is of prior importance, especially when these ones are not standardized. These tools are needed outside a specific HAL-SWH solution, these are components that can serve many scholarly infrastructures and institutions.

3.2.2 Ensuring metadata quality: the quest for more standardized practices

If guidelines alone are not enough to increase the adoption of good practices, then providing accurate tools is also insufficient. In other words, even if the form fields or drop-down menus are well-designed, they can't replace supporting actions.

Nurturing the moderators skills: towards a research software literacy

Although for a couple of decades, and especially due to the rise of data-driven research, librarians are used to describing types of documents far more malleable than publications, software specificities need to be explained to the teams in charge of the metadata curation. The aim is to enable moderators to spot which fields call for an extra-attention during the moderation phase and what is the required information to double check the metadata.

Moderators must be able to analyze the files that constitute the software, to find the required information and dialogue if necessary with depositors.

Even a minimalistic set of mandatory metadata to check necessitates some previous knowledge on software: not in order to be able to directly answer questions such as which license should be used, but rather to be aware that a software may be associated with several licenses or to be able to identify who the subject expert is within the institution. From a moderator's perspective, the point may be also to know that using the [SPDX License List](https://spdx.org/licenses/)³¹ helps enable efficient and reliable identification of such licenses. The purpose is different from implementing a consulting service dedicated to software.

Supporting the moderators: documentation and training

In collaboration with Software Heritage, the CCSD coordinates at the national level support actions for the HAL moderators, by providing:

- documentation for end-users³² and How To guide³³,
- documentation for the moderators³⁴,

³¹ <https://spdx.org/licenses/>

³² <https://hal.science/hal-01872189>

³³ <https://doc.hal.science/deposer/deposer-le-code-source>

³⁴ <https://inria.hal.science/hal-01876705>

- remote training sessions: at the end of the session, participants know which sources to use to check the metadata consistency. They learn how to use files from a code repository. They understand the difference between binaries and source code. They are able to check the SWHID type in case of SWHID deposit and to anticipate the impact of updates in their moderation workflow.
- in-person annual events that bring together HAL professional users (CasuHAL days)
- webinars of general reach about software in the scholarly ecosystem³⁵,
- a helpdesk via a dedicated mailing list

This centralized organization for documentation and training helps drive down the total cost of metadata capture. At a European level, the [Skills4Eosc workshops](#) may offer great opportunities: the network developed by the Skills4EOSC project brings together national and regional centers focused on the organization and transfer of knowledge in the fields of Open Science, FAIR data management and the European Cloud for Open Science (EOSC).

³⁵ <https://www.ccsd.cnrs.fr/2022/07/parlons-science-ouverte-les-logiciels-dans-hal/>

4 IMPLEMENTATION

4.1 Infrastructures: implementation of curation mechanisms

This chapter provides an overview of the implementation of curation mechanisms within various research software infrastructures, including scholarly repositories, publishers, and aggregators. The curation capabilities of these infrastructures are key to enabling high-quality metadata, ensuring that research software is effectively archived, described, and made accessible. The following subsections offer a snapshot of the mechanisms developed by FAIRCORE4EOSC partners for the infrastructures identified in the project proposal. These mechanisms outline the approaches taken to enhance the management and interoperability of research software, contributing to the broader goals of the European Open Science Cloud (EOSC). A detailed overview of the implementation of the SIRS report recommendations can be found in the SIRS GAP Analysis report (Azzouz-Thuderoz et al., 2023).

4.1.1 Scholarly repositories

InvenioRDM curation capabilities

InvenioRDM supports the creation and management of communities, which are collaborative spaces where research outputs can be curated and shared. Communities allow projects, institutions, domains, and events to manage their members, review submissions, and curate content to ensure high-quality research outputs. With the upcoming deployment of the connector between Software Heritage and Zenodo, the curated metadata updated during the community review is transferred to Software Heritage alongside the software submission.

Any user of an InvenioRDM instance can submit their record for review by a community. Community curators then review the metadata and file content using a dedicated interface that facilitates communication with the submitter. This interface allows both parties to edit the submission and discuss additional information or corrections, ensuring that metadata is accurate and comprehensive. Such changes may include correcting licensing information, refining subjects and keywords for better discoverability, and ensuring that the metadata includes links to external objects (journal articles, datasets, documentation, etc.). Community owners can also specify content policies and community extensions to streamline the review process and provide extra metadata.

InvenioRDM curation capabilities could be enhanced with the following information:

- DOI and Handle PIDs can be generated to refer to the records in an InvenioRDM instance.
- The communities can define their own "community extensions" that allow them to add extra metadata information to the records in InvenioRDM.

An example of more detailed documentation on managing communities in InvenioRDM can be found on the [Zenodo Help pages](#).

4.1.2 Publishers

Episciences curation capabilities

Episciences Link submission process with research software. In the context of the FAIRCORE4EOSC project, the Episciences team has implemented a frame in the article submission interface to link between the publication and the related research source code.

Episciences, a platform that hosts overlay journals published under the diamond open-access model, currently supports 35 journals. The new service allows authors, during the preprint submission process, to include a direct link to the source code associated with their research. This is made possible through the integration of the Software Hash ID (SWHID), which ensures that the source code is archived and accessible via Software Heritage. Episciences employs the same iframe technology as HAL to display the linked source code.

Each journal hosted on the Episciences platform can choose to enable this service based on its relevance to the journal's focus. For journals that opt to activate this feature, an additional review workflow can be introduced, allowing reviewers to request modifications to the source code or its metadata to enhance its quality.

Reviewers can access a submission and view the associated software artifacts within the same interface as the submission metadata and article. This is made possible through the use of an iframe, which allows them to easily browse the code. This technology provides a user-friendly experience for both researchers and reviewers.

Dagstuhl Curation Capabilities

As part of the FAIRCORE4EOSC project, Dagstuhl Publishing has significantly expanded both its submission system DSUB and its publication server DROPS to support supplementary material (especially research software) associated with scholarly articles (either contributions to journals or conference proceedings).

The submission system supports authors, editors and the publisher during the preparation of scholarly articles for publication. While submitting the peer-reviewed, camera-ready versions of scholarly articles, authors are asked for related supplementary materials such as research software in the now extended workflow. To ensure correct citation of the research data, authors are requested to provide not only the

URL of the material but also descriptive metadata (according to the CodeMeta standard) including SWHID. If the software is not yet archived on Software Heritage, authors are invited to request its archiving. The system also provides support in the form of pre-filled forms for the metadata. Metadata, if already available, is automatically read from the hosting platform, such as GitHub, or other sources, such as citation.cff files, and presented to the author in the form for revision and approval. Additionally, if desired by the author, information from the authors/contributors can be synchronized with the information in the corresponding scholarly article. The metadata collected during this process is then integrated into the scholarly article by the publisher during preparation for publication to ensure correct referencing and citation.

In addition, the publishing team carefully checks the metadata for completeness and accuracy. This curation process involves verifying critical metadata such as affiliations and funding details to ensure it is accurate and up to date. If there are any inconsistencies or missing information, the publishing team will make the necessary adjustments to the metadata. Authors are then given the opportunity to review the updated metadata as well as the reference immediately before publication and adjust it again if necessary. This curation process ensures that the metadata associated with both the scholarly article and its supplementary materials are of high quality and properly integrated into the publication. After publication, the reference/citation data is also part of the published metadata of the scholarly article.

As a next step, the metadata will not only be used in the scholarly article but will also be published on the DROPS publication server, if the metadata has not been published elsewhere. Furthermore, the metadata published in this way will then also be transferred to Software Heritage for long-term availability.

4.1.3 Aggregators

SwMath curation capabilities

The swMATH team has developed a curation interface in Python for the metadata editor curation process. Once an article about new software is published and indexed into zbMATH, the administrator of the swMATH database creates an entry and manually curates the findable associated metadata. The swMATH team is the authority that maintains the database and can process the necessary changes whenever they are needed.

swMATH is currently working to expose the swMATH metadata into the Datacite and Codemeta vocabulary. swMATH has based its identifier policy on two identifier types: intrinsic identifiers and extrinsic identifiers. While swMATH has its homemade identifier to index the software metadata, swMATH uses the SWHID as an indexer tool for the source code associated with the metadata.

We distinguish two types of articles that are related to software. The first category is the articles publishing a new contribution to the software. The administrator of the database curates them.

The second category is about articles citing software. Mainly, the zbMATH infrastructure processes SQL requests to spot keywords associated with software to find software mentions in the zbMATH articles automatically.

With the help of an internal tool to edit research articles, editors can identify the software mentioned in the two configurations described before, whether it is software cited in articles or a contribution to a new software presented in the article.

The swMATH metadata are harvestable through the zbMATH Open API³⁶, which publicly exposes all the zbMATH Open metadata, including the swMATH metadata.

DataCite curation capabilities

DataCite offers robust curation capabilities to enhance the quality, discoverability and usability of research outputs. This includes:

- Comprehensive [metadata schema](#) that includes mandatory, recommended and optional fields
- [Controlled vocabularies](#) for various properties
- Tools and validation services to check for completeness, consistency and adherence to standards
- Services to manage and update metadata records to facilitate ongoing maintenance and improvement of metadata quality
- Enriched metadata and ability to connect to related research outputs to improve discoverability
- [Discovery tools](#) to search and visualize metadata and connections
- Documentation, webinars, community meetings to educate and support metadata management and curation activities
- [Metadata Working Group](#) which consists of a group of metadata experts to develop and refine DataCite metadata schema with the wider community

DataCite metadata schema allows the community to register DOIs for [Software](#). Currently, there are over [500k software outputs](#) registered with DataCite. We are in the process of proposing an extension to the DataCite metadata schema to include SWHID as a controlled vocabulary type in the [relatedIdentifierType](#) property to enable connections to SWHIDs.

³⁶ <https://ems.press/journals/mag/articles/13614036>

4.2 Metadata management and curation

4.2.1 Metadata landscape overview

This chapter provides a brief overview of the metadata standards and vocabularies relevant to the curation and management of research software. The aim is to offer a non-exhaustive reference list to the various schemas and standards that support the description, citation, and sharing of software within the research community.

General metadata standards:

DataCite: A widely used standard for citing datasets, including software.

- [DataCite Schema](#)
- [Mappings for Software Citation](#)
- [CodeMeta mapping to DataCite](#)

Dublin Core: A general metadata standard used across various domains, including software.

- [Dublin Core Terms](#)

CrossRef: Another essential standard for registering Digital Object Identifiers (DOIs) for scholarly content, including software.

- [CrossRef Schema](#)

Schema.org: Provides a structured data format for web-based descriptions of software.

- [Schema.org SoftwareSourceCode](#)

Software specific schemas:

CodeMeta: A vocabulary specifically designed to describe software metadata.

- [CodeMeta Schema](#)

CITATION.cff: A format for adding citation metadata directly in software repositories.

- [CITATION.cff](#)

Biotools

- [Schema with software types](#)

The SWO ontology

- a resource for reproducibility in biomedical data analysis, curation and digital preservation [repository](#)

[SDO ontology](#),

[OntoSoft](#)

Infrastructures schemas:

- Scholarly repositories:
 - **Zenodo**
 - **Figshare**
- Repositories:
 - **GitHub**
 - **GitLab**
 - **Bitbucket**
- Registries/Aggregators:

- Astrophysics Source Code Library (ASCL)
- SwMath
- Research Software Directory
- Packaging Authorities:
 - PyPI
 - CRAN
 - npm

Other Relevant Standards:

- **Bioschemas:** Provides profiles for computational tools, particularly in the life sciences.
 - [Bioschemas Computational Tool Profile](#)
- **Computational Workflow**
<https://bioschemas.org/profiles/ComputationalWorkflow/1.0-RELEASE>

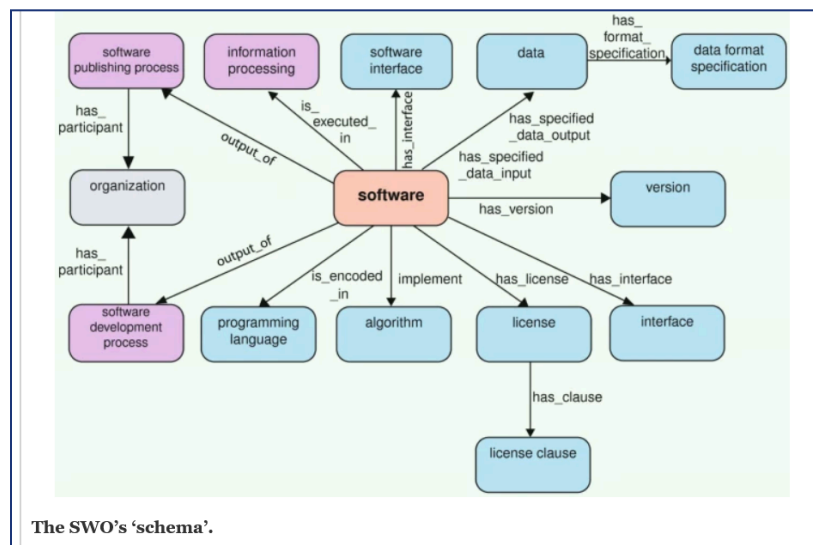


Figure 7 - The Software Ontology (SWO) landscape (Malone et al., 2014)

4.2.2 Controlled vocabularies

Controlled vocabularies are essential for ensuring consistency and clarity in the description and categorization of research software. This section provides an overview of key controlled vocabularies used in different aspects of research software management, including licensing, organizational identifiers, programming languages, and funding sources.

Licenses

Licenses define the terms under which software can be used, modified, and shared. Several controlled vocabularies are commonly used:

- **SPDX:** The Software Package Data Exchange (SPDX) list includes a wide range of licenses, covering open source, proprietary, and free software licenses. It is widely used by platforms like GitHub and GitLab.
 - [SPDX License List](#)
- **Creative Commons:** Provides licenses for creative works, including software documentation.
 - [Creative Commons Licenses](#)
- **OSI:** The Open Source Initiative (OSI) maintains a list of licenses that meet the Open Source Definition.
 - [OSI Approved Licenses](#)
- **FSF:** The Free Software Foundation (FSF) lists licenses that align with free software principles.
 - [FSF License List](#)
- **Government/Institutional Licenses:** Specific licenses tailored to government or institutional needs.
- Wikidata [software license class](#)

Organizational Identifiers

These vocabularies ensure the accurate identification of research organizations:

- [ORCID](#): Open Researcher and Contributor ID, unique identifiers for people.
- [ROR](#): The Research Organization Registry provides unique identifiers for research institutions.
- [GRID](#): The Global Research Identifier Database offers another set of identifiers for organizations.
- [FundRef](#): Tracks funding organizations, managed by CrossRef.
- [ISNI](#): Identifies organizations and individuals in research and creative activities.

Programming Languages

Identifying the programming languages used in software:

- [GitHub Linguist](#): A tool that helps classify programming languages in code repositories.

Development Status

Indicates the status of software development:

- [repostatus.org](#): Provides standardized terms like "active," "inactive," or "archived."

Runtime Platforms

Describes the environments where software can run, such as operating systems and application types.

Keywords/Subjects

Helps categorize software by content and scientific discipline:

- [EuroSciVoc](#): A vocabulary for categorizing scientific outputs in Europe.
- [OECD Field of Science](#): Classifies research by scientific discipline.

Funding

Tracks funding sources linked to research outputs:



FAIRCORE4EOSC has received funding from the EU's Horizon Europe research and innovation programme under Grant Agreement no. 101057264.

- [ROR](#): Also used for identifying funding organizations.
- [FundRef](#): Used to identify funding organizations.
- [OpenAIRE Projects](#) Links projects with their funding sources.

These controlled vocabularies help standardize how research software is described, making it easier to organize and share information across different systems.

4.2.3 Tools for metadata curation

In appendix E a tools table provides an overview of various tools available for managing and generating metadata for research software, particularly focusing on CodeMeta tools. Interoperability with the vocabularies list above, would be a welcomed improvement on CodeMeta tools. The list of tools isn't exhaustive. It includes their capabilities and functions, which range from generating metadata and supporting different formats to providing validation and conversion features. It helps researchers, developers, and information professionals understand the capabilities, limitations, and target users of each tool. The table also highlights the varying levels of support for different metadata formats, programming languages, and accessibility options, helping the community align its efforts with the FAIR principles by promoting interoperability, discoverability, and reuse of research software.

In this section, we will focus on the CodeMeta Generator and its potential to support the FAIR principles for research software.

The CodeMeta generator

Software Heritage maintains a tool for helping users to create the corresponding `codemeta.json` file, the CodeMeta generator³⁷. It consists of a simple form that users can fill in to generate a valid file. This file can then be added at the root of the software code repository.

Additionally, the generator now supports creating `codemeta.json` files in both v2.0 and v3.0 formats, and the form has been redesigned to include new functionalities, such as the Review box and role management.

The key functionalities include:

1. Import an already existing `codemeta.json` file (in v2.0 or v3.0). The form will be updated with the values found in the `codemeta.json` text area (where you pasted your file).
2. Validate json file
3. Generate file from form
4. Download file (to insert in code repository at root directory)

³⁷ <https://codemeta.github.io/codemeta-generator/>

[Generate codemeta.json v3.0](#) [Generate codemeta.json v2.0](#) [Reset form](#) [Validate codemeta.json](#) [Import codemeta.json](#) [Download codemeta.json](#)

Figure 8 - CodeMeta generator actions

The form has been reorganized, to include the new Review box.

Editorial review

Reference Publication

Review aspect

Review body

Figure 9 - The new Review box

Contributor #19

Given name

Family name

E-mail address

URI

Affiliation

Role
Start date:
End date:

Figure 10 - The new Role functionality

The Software Heritage metadata indexers

At Software Heritage, a system manages the storage and indexing of metadata to enhance software discoverability. This system handles two types of metadata: intrinsic metadata, which is stored with the code, and extrinsic metadata, which is archived separately. A translator converts the metadata into CodeMeta format for indexing, and the processed metadata is then stored in the Indexed Metadata Storage. For more details, refer to [Software Heritage's documentation](https://docs.softwareheritage.org/devel/swh-indexer/metadata-workflow.html)³⁸. A diagram in figure 11 below outlines the full workflow, showing the steps for collecting, preserving, indexing, and sharing metadata.

³⁸ <https://docs.softwareheritage.org/devel/swh-indexer/metadata-workflow.html>

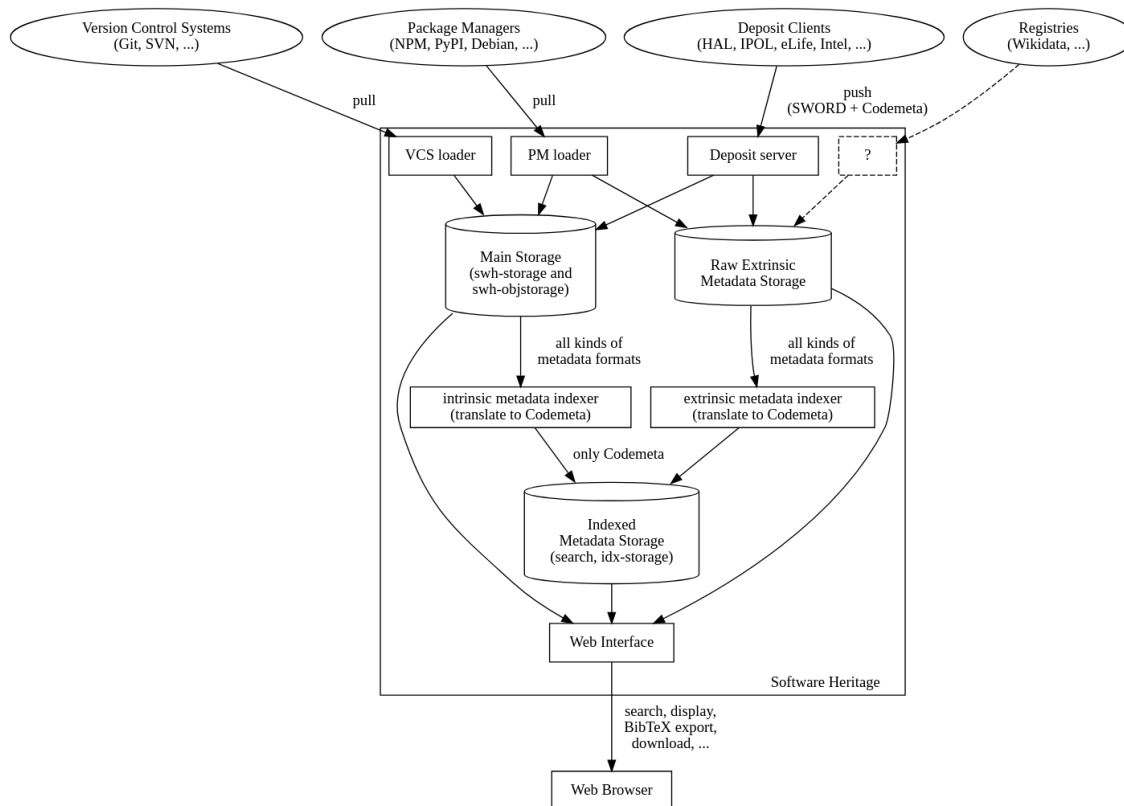


Figure 11 - Metadata workflow in Software Heritage following the metadata architecture³⁹

Simple Web-service Offering Repository Deposit (SWORD protocol) is an interoperability standard for digital repositories to accept and submit deposits of content from multiple sources. It is used in Software Heritage to connect with external deposit partners, such as HAL, eLife, IPOL.

³⁹ <https://docs.softwareheritage.org/devel/architecture/metadata.html>

5 THE CURATION & STANDARDISATION ROADMAP

5.1 Action steps for moving forward within EOSC and beyond

5.1.1 The challenges

The landscape of research software is intricately linked to a vast and diverse software ecosystem, making it challenging to ensure consistent practices across different platforms and disciplines. Supporting open infrastructures that serve both research software and the researchers who develop and use it is critical. Institutions must identify and promote key roles that support researchers in making their software open source, FAIR, citable, and reproducible. However, achieving high-quality metadata and rigorous peer-review of software artifacts comes at a significant cost, both in terms of time and resources. Additionally, there is a pressing need for a cultural shift within the research community to recognize software not just as a tool but as an essential component of research, an outcome of the research process, and a research object in its own right.

5.1.2 Concrete steps for the EOSC community

The EOSC community has made significant progress in developing connected infrastructures through the FAIRCORE4EOSC project and in identifying ways to improve software quality via the dedicated Infrastructures for Quality Research Software Task Force. However, challenges remain, and it is crucial for the academic community to allocate the necessary financial and human resources to support the operational, development, and training needs required for effective curation and the adoption of community standards.

Here is a suggested roadmap that infrastructures, institutions, funders and other stakeholders could consider to further support the scientific community:

Short-term (0-2 years)

1. **Support and training:**

Academic institutions should invest in curation and training activities for researchers and support staff, acknowledging the significant effort required.

2. **Infrastructure curation capabilities:** Infrastructures encompass a wide range of platforms, including aggregators, publishers and scholarly repositories.

- Infrastructures should invest in developing and implementing curation capabilities and functionalities aligned with community standards for research outputs, with specialized features tailored to Research Software.

- Curation capabilities should include the identification of automation requirements, to reduce the curation effort from end users.

3. Adopting and adapting metadata guidelines:

Institutions should require infrastructures to provide metadata capabilities that align with community-based guidelines, such as the FAIR-IMPACT Research Software MetaData (RSMD) guidelines, CodeMeta, and/or CFF metadata standards.

4. Community effort:

Infrastructures should actively engage in community-driven efforts (e.g., the SciCodes consortium) to develop and implement standards, guidelines, and best practices. By collaborating with the broader research community, infrastructures can contribute to and benefit from collective knowledge and resources, ensuring that their services are aligned with the evolving needs of the scientific community and supporting the advancement of open science.

5. Recognition & acknowledgement:

- **Career evaluation:**

Institutions should integrate curated metadata records into activity reports to highlight software as a recognized and valued research output, using this information in the career evaluation of researchers as an incentive.

- **Citation standard:**

Strengthening the connection between researchers and their software outputs by using the BibTeX @software type in articles and ensuring that software is properly cited and credited in scholarly work.

Medium-term (2-5 years)

1. Adoption:

Institutions and funders should support the adoption of infrastructures that propose curation capabilities to ensure the maintenance and sustainability of these infrastructures.

2. Robust curation processes:

Stakeholders should implement robust curation processes utilizing the capabilities of infrastructures. This includes:

- Metadata curation by moderators, curators, archivists, or other information specialists, similar to the model used by HAL, to ensure the quality and reliability of curated software metadata.

- Peer review of software as part of the scholarly process, similar to the practices used by IPOL, rOpenSci, or pyOpenSci, to validate the quality of submitted software as a research output.

3. Automation:

- Automation should be employed wherever possible to streamline the curation process and reduce the manual burden on researchers and curators.
- New possibilities of automation should be explored to optimize curation workflows.

4. Interoperability:

- **Identifiers:**
Infrastructures should support both intrinsic (e.g., SWHID) and extrinsic (e.g., DOI) identifiers for software identification and ensure that Software Hash Identifiers (SWHID) are exposed for archived resources whenever possible.
- **Exposing metadata:**
The adoption of standard APIs is crucial for making metadata harvestable across different platforms, thereby enhancing interoperability.

5. Feedback mechanisms:

The research community should establish regular feedback channels, similar to those used in Open Source communities, to enable users to report issues and suggest improvements of functionalities and workflows to the infrastructures.

Long-term (5-10 years and beyond)

1. Monitoring:

Institutions should actively monitor the software production within their laboratories to ensure the quality, visibility, and impact of research outputs. Effective monitoring relies on the curation capabilities of the infrastructure in use, which must be equipped to track, manage, and preserve software throughout its lifecycle.

2. Maintenance of community standards:

Funders and institutions should provide monetary and human resources to continue maintaining community efforts, such as CodeMeta, CFF and SWHID.

3. Sustainability:

- **Define and communicate measures:**
Infrastructures should define and communicate their sustainability measures, including

governance, retention, and end-of-life policies, as suggested by the SciCodes best practices (Task Force on Best Practices for Software Registries et al., 2020) and POSI principles (Bilder G, Lin J, Neylon C, 2020).

- **Establish funding models:**
Funders should establish funding models to ensure sustainable support for projects, infrastructures, and training initiatives.
- **Foster institutional collaboration:**
Institutions should actively collaborate across national and international levels to share resources, knowledge, and expertise, enabling sustainable practices and reducing duplication of effort.

5.1.3 Call to action

The EOSC (European Open Science Cloud) should take action to support key infrastructures and initiatives for Open Science. In doing so, it should meet the objectives set out by the EOSC SRIA and the EOSC SIRS report, particularly regarding the recognition of software as a research output and the interoperability of scholarly repositories. This effort should demonstrate Europe's contribution to structuring an international ecosystem for open science and research software that is efficient, transparent, and resilient, serving the scientific community and society at large.

*“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)

Brinkman et al. highlights the need for a collaborative approach where policies set the framework, infrastructures provide the necessary tools, and the community drives the actual practices. To achieve this, there must be ongoing dialogue and collaboration between policy makers, infrastructure providers, and the research community to ensure that research software is properly curated, recognized, and valued as a vital component of the scientific process.

Standards, guidelines, and metrics cannot be implemented in isolation. Similarly, infrastructure alone cannot achieve the goals of open science without the active engagement and participation of the research community. This collective approach will ensure that the principles of open science are not only adopted but also effectively integrated into the daily practices of researchers and institutions, paving the way for a more open, accessible, and impactful scientific ecosystem.

REFERENCES

No	Description/Link
R1	https://www.zotero.org/groups/5682994/faircoreeosc_d./library
R2	Azzouz-Thuderoz, M., Del Cano, L., Castro, L. J., Dumiszewski, Ł., Garijo, D., Gonzalez Lopez, J. B., Gruenpeter, M., Schubotz, M., & Wolski, M. (2023). SIRS Gap Analysis Report. https://zenodo.org/records/10376006
R3	Bilder, G., Lin, J., & Neylon, C. (2020). The Principles of Open Scholarly Infrastructure. https://doi.org/10.24343/C34W2H
R4	Declaration on Research Assessment. (2013). San Francisco Declaration on Research Assessment. DORA. https://sfdora.org/read/
R5	Di Cosmo, R., Gruenpeter, M., & Zacchiroli, S. (2018). Identifiers for Digital Objects: The Case of Software Source Code Preservation. 1–9. https://doi.org/10.17605/OSF.IO/KDE56
R6	Di Cosmo, R. (2023, March). SWHID specification kickoff meeting. SWHID kick-off meeting, Online Conference. https://hal.science/hal-04121507
R7	Directorate-General for Research and Innovation (European Commission) & EOSC Executive Board. (2022). Strategic Research and Innovation Agenda (SRIA) of the European Open Science Cloud (EOSC). Publications Office of the European Union. https://data.europa.eu/doi/10.2777/935288
R8	Eglen, S., & Nüst, D. (2019). CODECHECK: An open-science initiative to facilitate sharing of computer programs and results presented in scientific publications. Septentrio Conference Series, 1. https://doi.org/10.7557/5.4910
R9	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
R10	Garijo, D., Arroyo, M., Gonzalez, E., Treude, C., & Tarocco, N. (2024). Bidirectional Paper-Repository Tracing in Software Engineering. Proceedings of the 21st International Conference on Mining Software Repositories, 642–646. https://doi.org/10.1145/3643991.3644876
R11	Granger, S., Gruenpeter, M., Monteil, A., Nivault, E., & Sadowska, J. (2022, October 26). Modérer un dépôt logiciel dans HAL: Dépôt source et dépôt SWHID. Inria ; CCSD ; Software Heritage. https://inria.hal.science/hal-01876705
R12	Gruenpeter, M., Sadowska, J., Nivault, E., & Monteil, A. (2022). Create software deposit in HAL. Inria ; CCSD ; Software Heritage. https://inria.hal.science/hal-01872189
R13	Gruenpeter, M., Granger, S., Monteil, A., Chue Hong, N., Breitmoser, E., Antonioletti, M., Garijo, D., González Guardia, E., Gonzalez Beltran, A., Goble, C., Soiland-Reyes, S., Juty, N., & Mejias, G. (2023). D4.4—Guidelines for recommended metadata standard for research software within EOSC. https://doi.org/10.5281/ZENODO.8199104

R14	Katz, D. S., & Barker, M. (2023). The Research Software Alliance (ReSA). Upstream. https://doi.org/10.54900/zwm7q-vet94
R15	Le Berre, D., Jeannas, J.-Y., Cosmo, R. D., & Pellegrini, F. (2023). Forges de l'Enseignement supérieur et de la Recherche Définition, usages, limitations rencontrées et analyse des besoins [Report]. Comité pour la science ouverte. https://doi.org/10.52949/34
R16	Malone, J., Brown, A., Lister, A. L., Ison, J., Hull, D., Parkinson, H., & Stevens, R. (2014). The Software Ontology (SWO): A resource for reproducibility in biomedical data analysis, curation and digital preservation. <i>Journal of Biomedical Semantics</i> , 5(1), 25. https://doi.org/10.1186/2041-1480-5-25
R17	Mayernik, M. S. (2016). Research data and metadata curation as institutional issues. <i>Journal of the Association for Information Science and Technology</i> , 67(4), 973–993. https://doi.org/10.1002/asi.23425
R18	Rios, F. (2018). Incorporating Software Curation into Research Data Management Services: Lessons Learned. <i>International Journal of Digital Curation</i> , 13(1), Article 1. https://doi.org/10.2218/ijdc.v13i1.608
R19	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. http://arxiv.org/abs/2012.13117
R20	Treloar, A., & Wilkinson, R. (2008). Rethinking Metadata Creation and Management in a Data-Driven Research World. 2008 IEEE Fourth International Conference on EScience, 782–789. https://doi.org/10.1109/eScience.2008.41
R21	van de Sandt, S., Nielsen, L. H., Ioannidis, A., Muench, A., Henneken, E., Accomazzi, A., Bigarella, C., Lopez, J. B. G., & Dallmeier-Tiessen, S. (2019). Practice meets Principle: Tracking Software and Data Citations to Zenodo DOIs (Version 1). arXiv. https://doi.org/10.48550/ARXIV.1911.00295

APPENDICES

Appendix A: SIRS Gap report analysis and actions

Table 7 - SIRS Gap report analysis and actions

Dimension	Associated stakes	Gap, threat	Action point	Comment
Archive	Enabling long-term access to software source code	Software source code not archived: software is out of the scope of the institutional repositories and national archives	Using the SWH API to ensure source code archiving	Long term access to software source code raises also governance stakes: <ul style="list-style-type: none"> - infrastructures should be built from stable existing open source software building blocks - the infrastructure should be hosted and run by a non-profit organization to avoid risk of proprietaristation
Archive	Intrinsic metadata should be created and stored according to recognised best practices in software development because they may provide information about the versioning, the relations with other outputs and other identifiers	Metadata describing software not archived	Using the SWH API to ensure metadata archiving	In addition, metadata should be available under a CC0 license
Reference	Improving reproducibility of research by pinpointing the	Lack of common identifiers for software: referencing and describing software are	Requiring using SWHIDs in software sustainability plans	

Dimension	Associated stakes	Gap, threat	Action point	Comment
	exact version of any software artifact via intrinsic identifiers	separate concerns. Inconsistent use of PIDs complicates the unique identification of software and the proper attribution of credits.	for project proposals	
Describe	Providing software descriptions to improve their citability and their discoverability	Lack of support to keep and maintain intrinsic metadata	Adoption of guidelines and common software metadata quality indicators is needed	Aggregated metadata should be available “as open as possible as closed as necessary” (e.g. to respect GDPR regulations)
Describe		Lack the adoption of a <i>common software metadata schema</i>	Encourage use of CodeMeta, funding for community and governance	
Describe		Aggregators rarely provide access to the metadata and data through an open API		
Credit	Software heavily relies on collaborative work and different types of contributions: coding, debugging, testing, designing, etc.	Lack of a standardized taxonomy for research software contributor roles to categorize contributions	Generalization of Contribution Role Ontology (CRO) and integration in Codemeta.	
Credit		No common software citation practice adopted in infrastructures: software citation practices are not standardized and very often, software is not considered as a citable output	Require adoption of standards like CFF.	

Dimension	Associated stakes	Gap, threat	Action point	Comment
Credit		No common software quality indicators and best practices for software metadata	Development of common software indicators, guides based on Codemeta	
Governance and sustainability		No adoption of SIRS recommendations from infrastructures	Require governance and sustainability to be part of the software management plan. Compliance with best community practices	
Publishers platforms	Adoption of SIRS recommendations by publishers		Working Group to synchronize and ease the adoption of community standards into JATS: see JATS4R recommendation for software https://jats4r.niso.org/software-citation/	For now, specialized publishers are the ones who have mostly changed their practices: support for software outside of specialist journals is still limited. But a general effort is required, to implement standardized practices, but it must be as acceptable as possible for all the stakeholders.

Appendix B: Description of Infrastructures Types from FAIR-IMPACT D4.4

Table 8 - 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)	<ul style="list-style-type: none"> • HAL • Zenodo • Dryad
Registries (catalogs)	“Research software registries are typically indexes or catalogs of software metadata, without any code stored in them; while in <i>research software repositories</i> , software is both indexed <i>and</i> stored (Lamprecht et al., 2020).” (Garijo et al., 2022)	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • OpenAIRE • swMATH.org
Software development platform / Forge	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.	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • OS: Debian, Ubuntu, WinGet

Infrastructure/ Platform type	Definition/ Why do we focus on this actor/infrastructure?	Examples
	software" https://doi.org/10.1007/s41019-017-0050-4	<ul style="list-style-type: none"> • OS independent: Conda, Homebrew • Python: PIP • R: CRAN

Appendix C: Infrastructure metadata curation capabilities evaluation

To effectively evaluate your infrastructure's capabilities for curating software metadata, it's essential to gather detailed information on the tools, processes, responsibilities, and standards that support this effort. The following tailored questionnaire is designed to assess these critical areas, providing insights into how well your current systems and practices align with the needs of software metadata curation. By focusing on these key aspects, this evaluation will help identify strengths, potential gaps, and opportunities for improvement in your software metadata curation infrastructure.

- Tools and interfaces:**
 - Which tool(s) are used for software metadata curation, and how do they support the process?
- Roles and responsibilities:**
 - Who is responsible for software metadata curation, and how are these roles defined?
- Metadata management:**
 - Who manages the software metadata database, and what measures ensure its accuracy and maintenance?
- Access control and authority:**
 - Who can modify software metadata, and what protocols are in place to maintain its quality?
- Standardisation and mapping:**
 - What standard vocabularies are used, and how are mappings to your home vocabulary maintained?
- Identifier Policy:**
 - What are your policies for assigning identifiers to software metadata and source code?
- Linking with Publications:**
 - How is software metadata linked to relevant publications, and how are contributions and citations distinguished?

Appendix D: Intrinsic vs. Extrinsic identifiers from FAIR-IMPACT D4.4

Identifiers, essential for linking research outputs, can be broadly classified into two types: extrinsic, which rely on a registry to associate the identifier with the object, and intrinsic, which are inherently tied to the object itself without requiring an external register.

Table 9 - Overview of the main differences between intrinsic and extrinsic identifiers⁴⁰

	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 type⁴¹	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 ⁴²	Parmap, Scikit-Learn, Coq, etc.
Type of PID	Identifiers for <u>D</u> igital <u>O</u> bjects	Digital Identifiers of <u>O</u> bjects
Technical challenges	Granularity of the object Software lifecycle Economic sustainability ⁴³	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)

⁴⁰ <https://www.softwareheritage.org/2020/07/09/intrinsic-vs-extrinsic-identifiers/>

⁴¹ For further information, see the diagram below : (Research Data Alliance/FORCE11 Software Source Code Identification WG)

⁴² sw.h:1:cnt:bb0faf6919fc60636b2696f32ec9b3c2adb247fe

⁴³ “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.”

	<i>Intrinsic identifiers</i>	<i>Extrinsic identifiers</i>
		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: <ul style="list-style-type: none"> • Researchers • Software developers • Research teams 	Depends on the use case, but mainly for: <ul style="list-style-type: none"> • Research institutions • Scholarly repositories • Publishers • Aggregators

Appendix E: The Episciences platform

Epijinfo Sandbox journal

[Home](#)
[About](#)
[Boards](#)
[For Authors](#)
[Volumes](#)
[Sections](#)
[Browse latest articles](#)
[Credits](#)
[Editorial Staff members](#)
[News](#)


[Browse by author](#)
[Abstracted and indexed in](#)
[Dashboard](#)
[Submit a document](#)
[My Account](#)
[Journal management](#)
[Statistics](#)

Antonin Chambolle ; Daniele de Gennaro ; Massimiliano Morini - Discrete-to-continuous crystalline curvature flows
epijinfo:1975 - [Pre-Production] Epijinfo sandbox

Go to the management page for this article

Software

References


Software Heritage

Navigating in archived swh:1:dir:76d091b28da372eb245ceac59e706f2e2ad178bf

Reset view
View in the archive

8bfd74 / CSCbox /

File	Mode	Size
amazon_graph		
codes		
third_party		
AUTHORStxt	-rw-r--r--	120 bytes
LICENSE	-rw-r--r--	31.6 KB
README.txt	-rw-r--r--	746 bytes
main_CSC.m	-rw-r--r--	5.3 KB

Add software

Linked publications - datasets - software

Author's comments / Cover letter

Rating

Evaluation

Scientific relevance

blablabla

Grade (coefficient 5) 0/10

Comment

B
I
U
↶ ↷
≡ ≡ ≡ ≡
⋮ ⋮



Figure 12 - Episciences platform



FAIRCORE4EOSC has received funding from the EU's Horizon Europe research and innovation programme under Grant Agreement no. 101057264.

Appendix F: Metadata tools in the Research Software landscape

Table 10 - Comparison of metadata tools for Research Software

Name & url/repository	Description	Repo	PID	Lang	Target user	Coverage CodeMeta & version	Helpers to complete form	Accessibility	License for reuse	Review result
Simple react form	Generate codemeta form on CodeMeta site link			javascript	researcher	not full	no	on chrome only	possible	no
codemetar	Generate codemeta for R packages; + generic codemeta manipulation		10.5281/zenodo.2648599	R	developer	2.0.0				
CodeMeta file generator	Generate codemeta			Ruby	developer	not full & 0.1.0	no			

Name & url/repository	Description	Repo	PID	Lang	Target user	Coverage CodeMeta & version	Helpers to complete form	Accessibility	License for reuse	Review result
Bolognese	Primarily a tool for conversion between formats provided by DataCite, including codemeta and schema.org		10.5438/n138-z3mk	Ruby	developer	1.0.0				
Codemeta Generator	A (client-side) web application to generate Codemeta documents (aka. codemeta.json). link			javascript	researcher	2.0 and 3.0	only licenses	run locally on browser	GNU AGPL	no
schema.org editor	Schema Generator link			Python 3.4	researcher	No CodeMeta	no	on browser (https://schema.pythonanywhere.com/SoftwareSourceCode)	possible	yes

Name & url/repository	Description	Repo	PID	Lang	Target user	Coverage CodeMeta & version	Helpers to complete form	Accessibility	License for reuse	Review result
cffinit	CFF initialization tool	link	10.5281/zenodo.1404735		researcher	not CodeMeta , 10 cff properties	no spdx list or other helpers	on browser		
cffconvert	CFF conversion tool		10.5281/zenodo.3338153						Apache Software License (Apache 2.0)	
codemetapy	Generate codemeta for Python package from existing pip information or scratch	link		Python	researcher			command line	possible (GPL)	
Google check generated schema										
ames	Convert codemeta to DataCite (with other metadata	link	10.2202/D1.1265	Python	researcher	2.0.0	no	command line	yes	yes

Name & url/repository	Description	Repo	PID	Lang	Target user	Coverage CodeMeta & version	Helpers to complete form	Accessibility	License for reuse	Review result
	transforms)									
convert_codemeta	Convert codemeta based on crosswalk	link		Python	researcher	2.0.0	no	command line	yes	yes
cite my code	Detects a CFF in a GitHub repo and generates BibTeX. Would be nice to get it fully working/working with CodeMeta files	link		Javascript	researcher	No CodeMeta	no	no	yes	no
json-ld validator		link				No CodeMeta	not for sw	yes	CC0	
	SchemaApp jsonld-schema-generator	link				No CodeMeta		browser	no	

Name & url/repository	Description	Repo	PID	Lang	Target user	Coverage CodeMeta & version	Helpers to complete form	Accessibility	License for reuse	Review result
eossr	CLI and web interface to validate CodeMeta schema. CLI and web interface to convert CodeMeta (codemeta.json) to Zenodo schema (.zenodo.json)	link	https://doi.org/10.5281/zenodo.1065238	Python	researcher	2.0.0		command line, gitlab CI, github action, browser	MIT	
somef	Software Metadata Extraction Framework	link	https://doi.org/10.5281/zenodo.10848711	Python	developer-researcher	2.0.0		run locally	MIT license	

Name & url/repository	Description	Repo	PID	Lang	Target user	Coverage CodeMeta & version	Helpers to complete form	Accessibility	License for reuse	Review result
Somesy	Tool to synchronise software metadata	link		Python	developer	2.0.0		run locally	MIT license	