

Making Research Software Visible, Citable, and Preserved: A Metadata Deep Dive for RSEs

Morane Gruenpeter, Director of Scholarly Ecosystem



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

What is software?



The Treachery of Images, René Magritte

Software as a concept

- **project** or entity
- the **community** around the project
- the software **idea** / algorithms / solutions

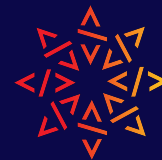
Not a digital artifact

Software artifacts

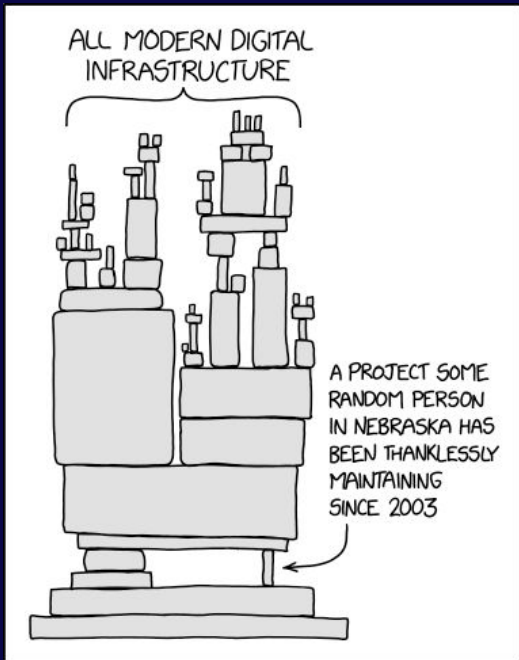
- Executables
 - For multiple environments
- **Source code**

A very large collection of digital artifacts

Source code is...



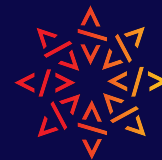
The invisible backbone of all digital infrastructure



<https://xkcd.com/2347/>

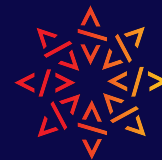
Preserving it is crucial

Goals



- ★ **Introduction: what is at stake?**
- ★ Building the software pillar of **Open Science**
- ★ Deep dive for **#BetterMetadata**
 - using the **RSMD Guidelines**
- ★ **Supporting research: new players in Academia**

What is at stake?



Archive

- make sure we can access to retrieve the software (reproducibility)

Reference

- make sure we can identify the software artifacts (reproducibility)

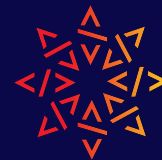
Cite (for credit)

- make it rewarding to create software by giving credit to authors (evaluation!)

Describe

- make it easy to discover the software projects (visibility)

A universal source code archive



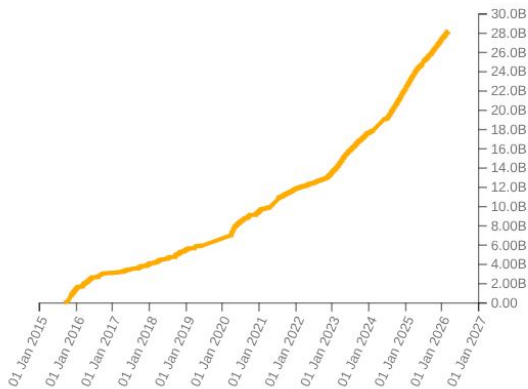
Collect, preserve and share all software source code



Preserving our heritage, enabling better software and better science for all

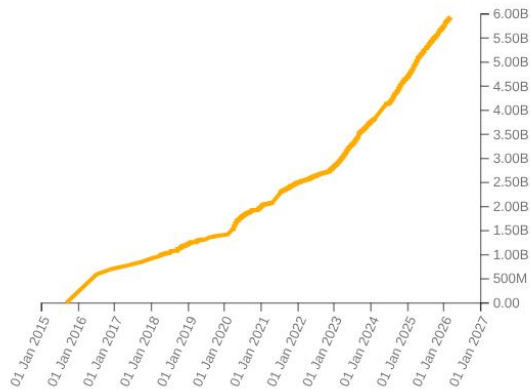
Source files

28,148,748,800



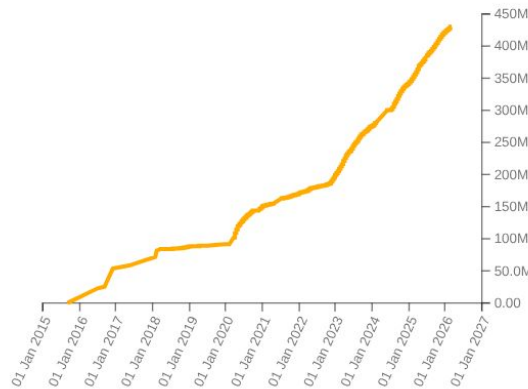
Commits

5,920,096,480



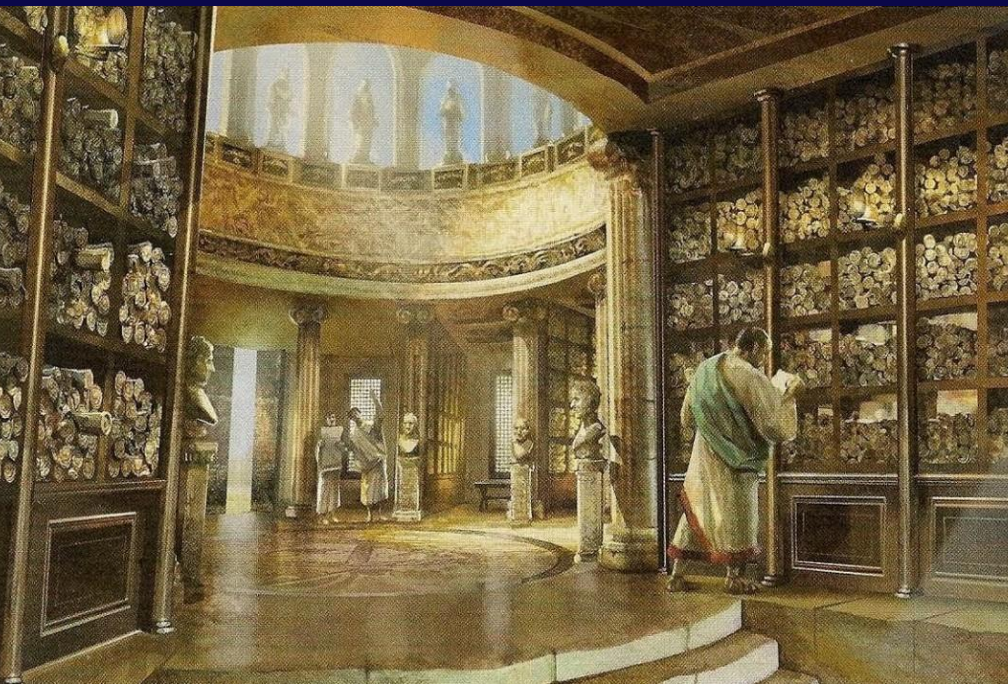
Projects

429,963,770



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

The library of Alexandria of source code



In Collaboration with



unesco

Launched in 2016 by

Inria

Hosted SYWH Foundation by

Inria

FONDATION

The 2025-2026 Sponsors

Diamond sponsors



Platinum sponsors



Gold sponsors



Silver sponsors

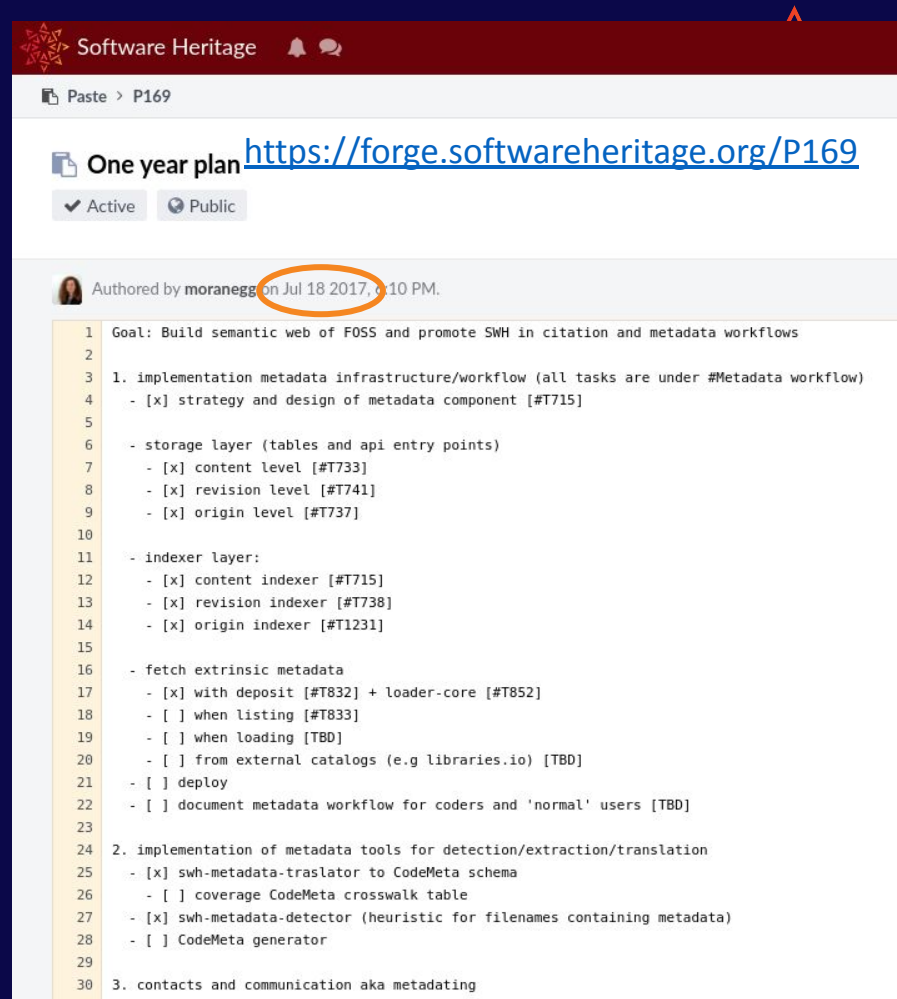


Bronze sponsors



Building the semantic web of FOSS

1. Analyze the metadata landscape
 - a. Discovering CodeMeta
2. Choose a vocabulary (DON'T create a new one)
3. Develop components:
 - a. Get or receive metadata
 - b. Find metadata
 - c. Store metadata
 - d. Translate metadata
4. Communicate - metadating - Raising awareness about the importance of Metadata
5. Identify the users



The screenshot shows the 'One year plan' project page on the Software Heritage website. The page header includes the 'Software Heritage' logo and navigation icons. The project title is 'One year plan' with a link to <https://forge.softwareheritage.org/P169>. Below the title are buttons for 'Active' and 'Public'. The author is 'moranegg' with a profile picture, and the creation date is 'Jul 18 2017, 8:10 PM'. The main content is a list of tasks for building a semantic web of FOSS, organized into three sections: 1. implementation metadata infrastructure/workflow, 2. implementation of metadata tools for detection/extraction/translation, and 3. contacts and communication aka metadating. The tasks are listed with checkboxes and associated issue numbers.

Software Heritage

Paste > P169

One year plan <https://forge.softwareheritage.org/P169>

Active Public

Authored by moranegg on Jul 18 2017, 8:10 PM.

```
1 Goal: Build semantic web of FOSS and promote SWH in citation and metadata workflows
2
3 1. implementation metadata infrastructure/workflow (all tasks are under #Metadata workflow)
4   - [x] strategy and design of metadata component [#T715]
5
6   - storage layer (tables and api entry points)
7     - [x] content level [#T733]
8     - [x] revision level [#T741]
9     - [x] origin level [#T737]
10
11   - indexer layer:
12     - [x] content indexer [#T715]
13     - [x] revision indexer [#T738]
14     - [x] origin indexer [#T1231]
15
16   - fetch extrinsic metadata
17     - [x] with deposit [#T832] + loader-core [#T852]
18     - [ ] when listing [#T833]
19     - [ ] when loading [TBD]
20     - [ ] from external catalogs (e.g libraries.io) [TBD]
21     - [ ] deploy
22     - [ ] document metadata workflow for coders and 'normal' users [TBD]
23
24 2. implementation of metadata tools for detection/extraction/translation
25   - [x] swh-metadata-traslator to CodeMeta schema
26   - [ ] coverage CodeMeta crosswalk table
27   - [x] swh-metadata-detector (heuristic for filenames containing metadata)
28   - [ ] CodeMeta generator
29
30 3. contacts and communication aka metadating
```

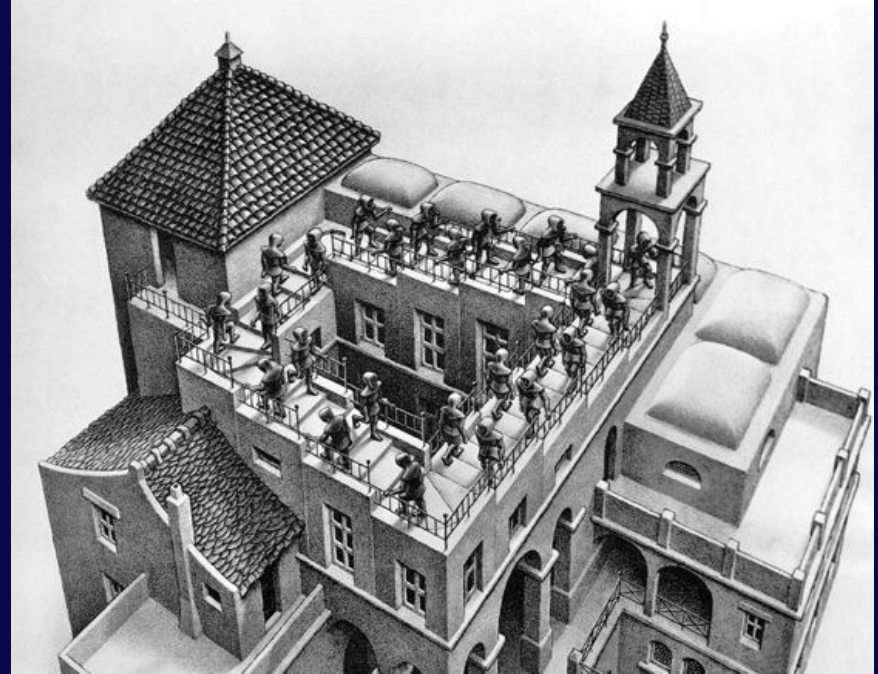


Who is responsible for the MetaData?

The Metadata is a tool for:

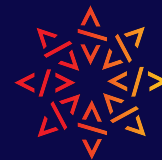
- Recognition
- Visibility
- Findability
- Understanding
- Reuse / Re-execute

Without the Metadata we lose the key to decipher the knowledge.



Ascending and Descending by M. C. Escher

Goals



- ★ **Introduction:** what is at stake?
- ★ **Building the software pillar of Open Science**
- ★ Deep dive for **#BetterMetadata**
 - using the **RSMD Guidelines**
- ★ **Supporting research:** new players in Academia

A pillar in Open Science: Software facets in Research



Research Software

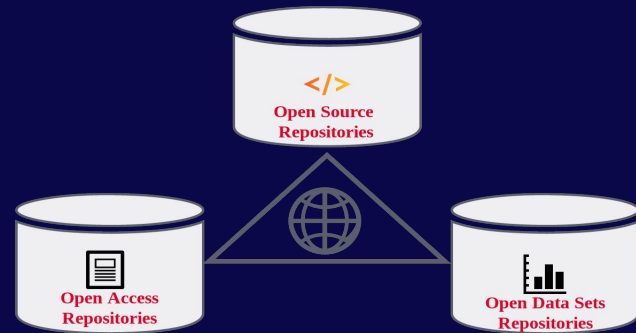
→ created

- during the research process
- for a research purpose

Software in research

→ used for research

FAIR4RS output: Gruenpeter et al. Defining Research Software: a controversial discussion (Version 1). Zenodo. <https://doi.org/10.5281/zenodo.5504016>



*Three pillars of Open Science
Software Heritage CC-BY 4.0 2019*

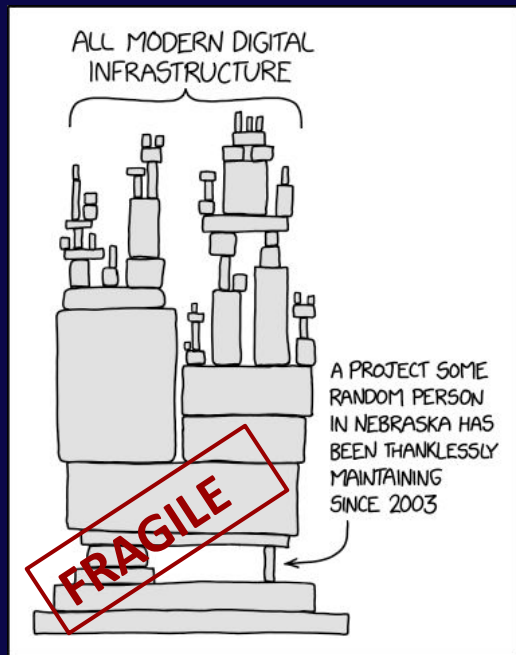
Software has multiple facets:

- a **tool**
- a research **outcome** or result
- **the object** of research

The software stack: More than Research Software

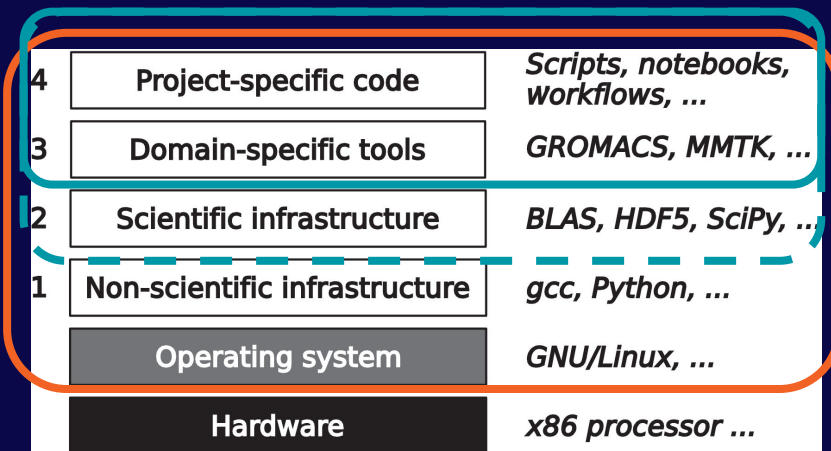


Research Software is a thin layer on top of the global software stack



<https://xkcd.com/2347/>

What's the the Software Collapse?

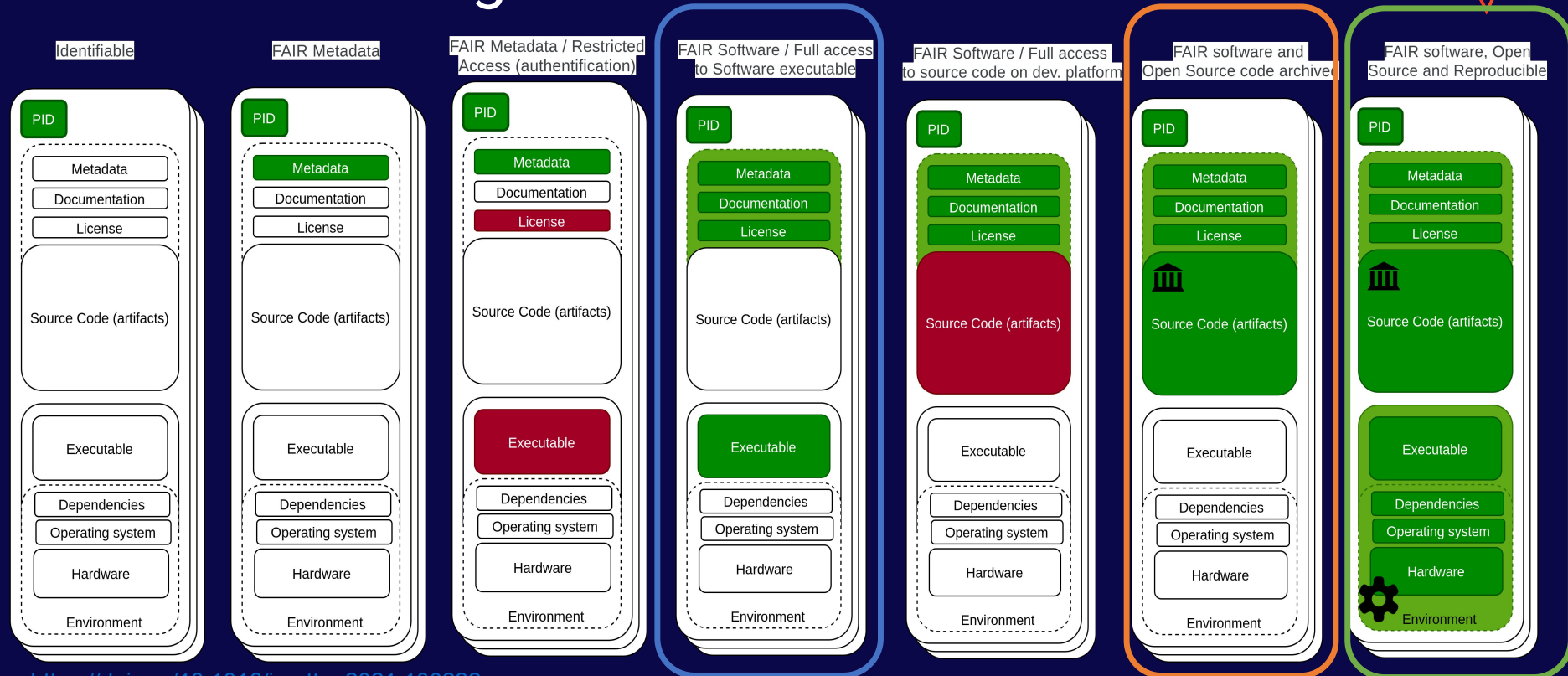


Konrad Hinsén. Dealing With Software Collapse. Computing in Science and Engineering, 2019, 21 (3), pp.104-108. [\(10.1109/MCSE.2019.2900945\)](#). [\(hal-02117588\)](#)

Can be preserved in a:

- [Scholarly Infrastructure](#)
- [Universal Source Code Archive](#)

FAIR is not enough for Software: is CodeMeta?



<https://doi.org/10.1016/j.patter.2021.100222>

FAIR4RS published in 2022

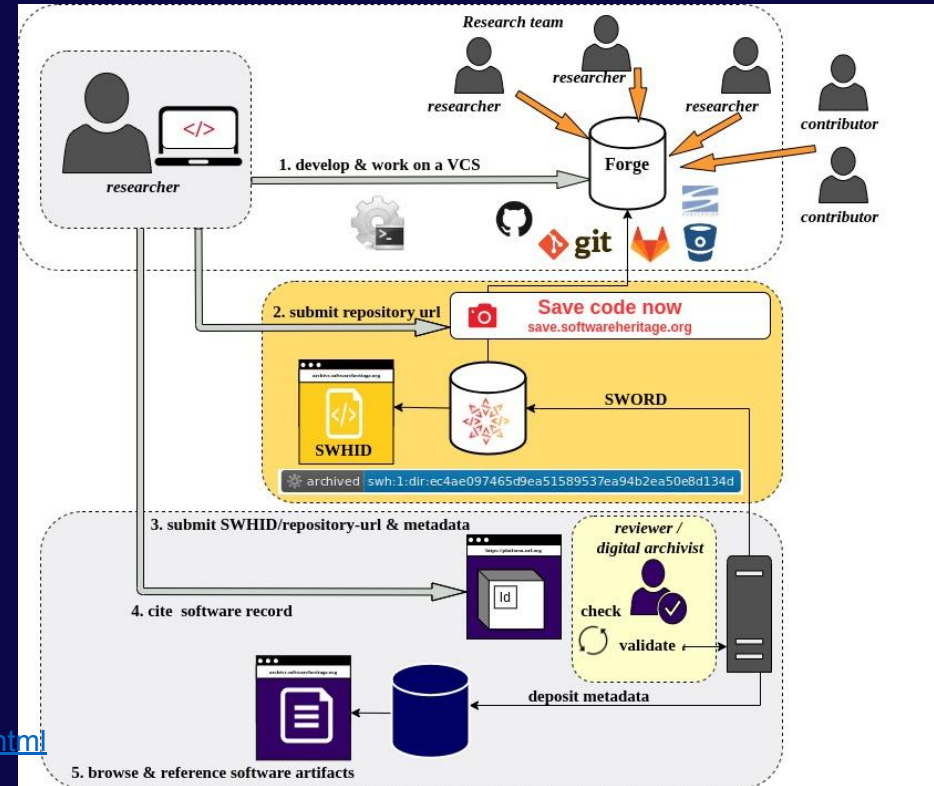


CodeMeta in the software deposit workflow at SWH

2 steps process:

- Save code now: Public Forge URL
- Metadata deposit on existing asset in the archive using CodeMeta

Can be used by all types of infrastructures

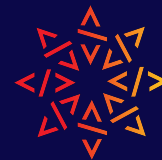


Documentation Used

<https://docs.softwareheritage.org/user/deposit/index.html>

<https://docs.softwareheritage.org/devel/swh-deposit/api/index.html>

Goals



- ★ **Introduction:** what is at stake?
- ★ Building the software pillar of **Open Science**
- ★ **Deep dive for #BetterMetadata**
 - **using the RSMD Guidelines**
- ★ **Supporting research:** new players in Academia



Where is the metadata available?

Software development platforms (on page)

- GitHub
- Bitbucket
- SourceForge
- ...

Package managers

- PyPI
- NPM
- ...

Intrinsic metadata

In the source code (as a file)

- README
- LICENSE
- AUTHORS
- SBOM
- Package manager file
- **codemeta.json / CFF file**
- ...

Scholarly repositories

- Zenodo (InvenioRDM)
- HAL
- Dataverse
- ...

Scholarly publishers (on record)

- IPOL
- eLife
- Dagstuhl
- Episciences
- ...

Catalogs, registries, aggregators

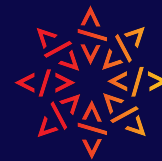
- ASCL
- CORE
- OpenAire
- OpenAlex
- [libraries.io](#) => Ecosyste.ms
- RSD - eScience center
- swMath
- ...

Citation / Mentions in Research Articles

- Softcite - extracting 3 types
 - Created
 - Used
 - Shared

Intrinsic metadata

In the *software source code* itself



- README
- LICENSE
- AUTHORS
- **codemeta.json**
- package management
 - pom.xml
 - package.json
 - ...
- CITATION.cff
- .About
- ...

```
1 {
2   "@context": "https://doi.org/10.5063/schema/codemeta-2.0",
3   "@type": "SoftwareSourceCode",
4   "license": "https://spdx.org/licenses/LGPL-2.0-only",
5   "codeRepository": "git+https://github.com/rdicosmo/parmap.git",
6   "datePublished": "2011-07-18",
7   "dateModified": "2022-01-03",
8   "issueTracker": "https://github.com/rdicosmo/parmap/issues",
9   "name": "Parmap",
10  "version": "1.2.5",
11  "applicationCategory": "Parallel computing",
12  "developmentStatus": "active",
13  "referencePublication": "https://doi.org/10.1016/j.j.procs.2012.04.202",
14  "programmingLanguage": [
15    "OCaml"
16  ],
17  "operatingSystem": [
18    "Linux",
19    "MacOS"
20  ],
21  "relatedLink": [
22    "https://opam.ocaml.org/packages/parmap/"
23  ]
24 }
```

Human readable (e.g README)

Machine actionable (e.g codemeta.json)

CodeMeta: A Rosetta Stone for Software Metadata



The CodeMeta Project

- A subset of schema.org
- An academic community and a community-led [governance model](#)
- An interoperable MD standard
- 10 years in the making (a few dates)
 - ◆ 2015 - 2016 FORCE11 [Software Citation WG](#)
 - 2016 The [future of Metadata workshop](#) (Poster)
 - ◆ 2017 - 2023 FORCE11 [Software Implementation WG](#)
 - 2020-2021 CodeMeta Task Force
 - ◆ 2020 Recommended by the [EOSC SIRS report](#)
 - ◆ 2021 Launch of the [SciCodes consortium](#) for infrastructures
 - ◆ 2022 Setting up governance the [CodeMeta PMC](#)
 - ◆ 2023 Recommended by the [FAIR-IMPACT RSMD](#)
 - ◆ 2025 Cross-integrations in FAIRCORE4EOSC and launch of [SSSOM evolution](#)
 - ◆ 2025 First [CodeMeta Party](#) preparing the v4.0

**Created and maintained
by the community**



SciCodes

Consortium of
scientific
software
registries and
repositories

CodeMeta tools

- An open source tool to create codemeta.json files
 - Use it directly on the CodeMeta [hosted version](#)
 - Contributions are welcome on the [code repository](#)
 - Created and maintained by



Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

The Auto-CodeMeta-Generator: <https://autocodemeta.linkeddata.es/>

CodeMeta generator



Most fields are optional. Mandatory fields will be highlighted when generating Codemeta.

The software itself

Name

the software title

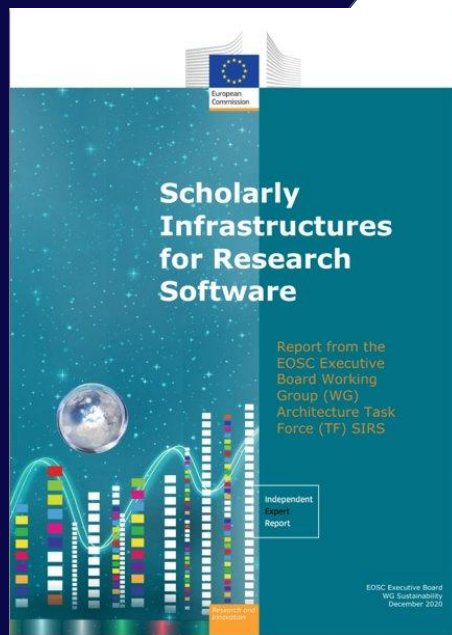
Description

Creation date

First release date

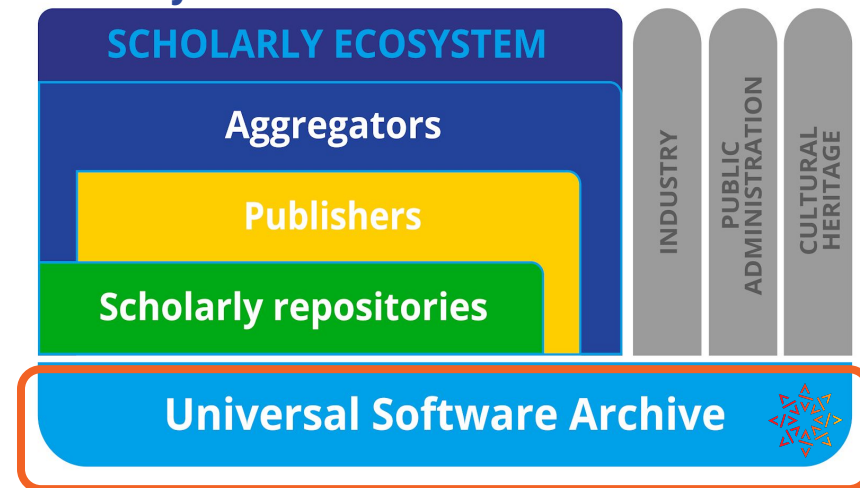


Cross-Infrastructure Coordination Starts with Clear Policy Decisions



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

Scholarly Infrastructures for Research Software (SIRS)



Short term recommendations

- Strengthening interactions between
 - Aggregators, publishers, scholarly repositories, and Software Heritage
- Metadata standards & tools
- Generalizing the use of PIDs (extrinsic & intrinsic) => **SWHID**



ISO/IEC 18670:2025

Software Heritage as a Metadata-broker



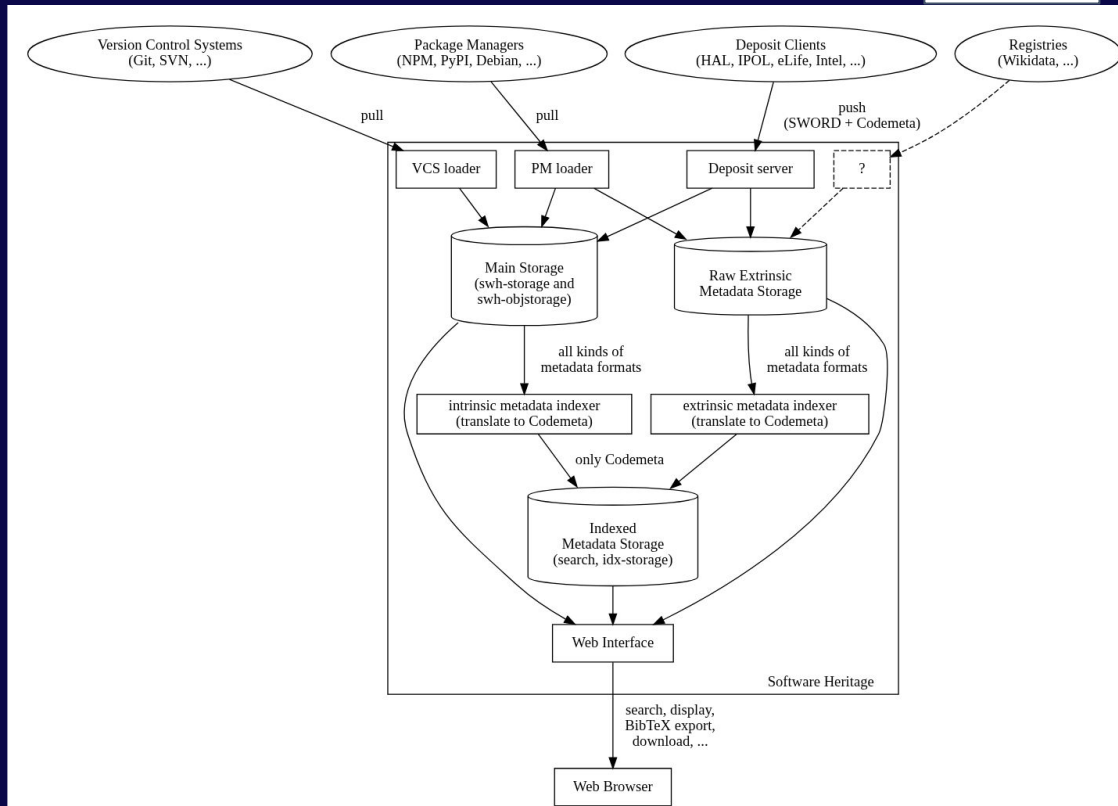
Leveraging the CodeMeta vocabulary

- A subset of **schema.org**
- Community governed project

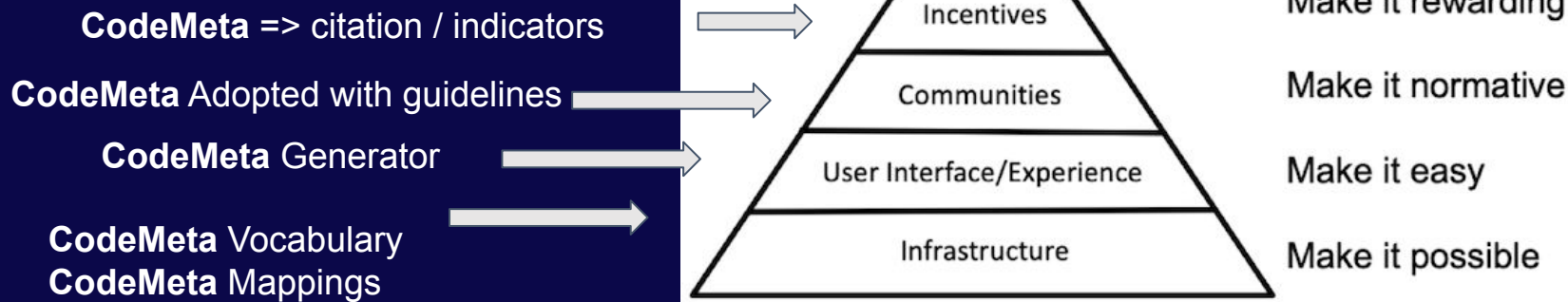
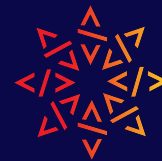
The CodeMeta Project

Software Heritage has the capacity to

- Ingest metadata
- Extract metadata
- Deliver consolidated views to stakeholders



A culture shift: Making **software** a first class research output



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

CodeMeta - More than a few added properties to schema.org (#453)

The Research Software MetaData guidelines

The RSMD seven Aspects

1. General Metadata Requirements

2. Accessibility & preservation

3. Reference & identification

4. Description & classification

5. Attribution & credit

6. Reuse, licensing & legal aspects

7. Re-execute: Dependencies & execution environment

SIRS report

FAIR4RS

A = Archive

A = Accessible

R = Reference

F = Findable

D = Describe

I = Interoperable

C = Cite

R = Reusable



<https://github.com/FAIR-IMPACT/RSMD-guidelines>
<https://fair-impact.github.io/RSMD-guidelines/>

Each aspect has a high-level objective with a series of recommendations

High-level objective

Actionable, detailed recommendations

Description & classification		https://github.com/FAIR-IMPACT/RSMD-guidelines
Objective		
Objective: Software is properly described with name, purpose and functionalities alongside other software specific metadata properties (programming language, domain, etc.) to ensure software findability.		
ID	Recommendation	Priority
RSMD-4.1	Add software name and description of the software's functionality and purpose, using a README file in the root directory of the source code or other intrinsic metadata file (e.g codemeta.json with the properties <i>name</i> and <i>description</i>).	Essential ☆☆☆
RSMD-4.2	Add descriptive metadata for classification purposes on metadata record (extrinsic metadata), which can be available in a scholarly infrastructure. This includes, but is not limited to: <ul style="list-style-type: none">• Name• Description• Domain• Programming language• Date created	Essential ☆☆☆

Objective:

To ensure the collection, curation, and maintenance of research software metadata, the following general requirements are recommended for end users, including researchers, software engineers, curators, and institution staff.

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

#RSMD_cheklist

1. General Metadata Requirements

2. Accessibility & preservation

3. Reference & identification

4. Description & classification

5. Attribution & credit

6. Reuse, licensing & legal aspects

7. Re-execute: Dependencies & execution environment

Objective:

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

- ❑ Is the software record or/and are the software artifacts accessible and preserved?
 - ❑ Is the software source code preserved in the SWH universal source code archive, Software Heritage? (RSMD-2.1)
 - ❑ Is the software archived in a scholarly repository (e.g Zenodo, HAL)? (RSMD-2.2)
 - ❑ Can it be accessed and downloaded?

- ❑ Is the software registered in a disciplinary or community registry (e.g ascl.net, bio.tools, swMath, RRID portal, RSD, WikiData, DataCite) (RSMD-2.3)
 - ❑ Can it be found in a search engine?

1. General Metadata Requirements

2. Accessibility & preservation

3. Reference & identification

4. Description & classification

5. Attribution & credit

6. Reuse, licensing & legal aspects

7. Re-execute: Dependencies & execution environment

Objective:

To ensure that research software projects, modules, versions and source code artifacts can be precisely identified and referenced.

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

1. General Metadata Requirements

2. Accessibility & preservation

3. Reference & identification

4. Description & classification

5. Attribution & credit

6. Reuse, licensing & legal aspects

7. Re-execute: Dependencies & execution environment

Objective:

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

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

#RSMD_cheklist

1. General Metadata Requirements

2. Accessibility & preservation

3. Reference & identification

4. Description & classification

5. Attribution & credit

6. Reuse, licensing & legal aspects

7. Re-execute: Dependencies & execution environment

Objective:

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

- ❑ Are the authors & contributors identified and acknowledged?
 - ❑ Is the author's information available in the source code as intrinsic metadata? (RSMD-5.1)
 - ❑ Is the author's information available on the metadata record as extrinsic metadata? (RSMD-5.2)
 - ❑ Is the list of authors exhaustive or is a collective author used? (RSMD-5.8)
- ❑ Are people identifiers used? (ORCID, ID-HAL, ID-REF, etc.) (RSMD-5.3)
- ❑ Are the roles of the authors and contributors specified? (RSMD-5.4)
- ❑ Is a citation preference provided? (RSMD-5.5)
- ❑ Is software explicitly cited and included in the bibliography (RSMD-5.6)?
- ❑ If applicable, in the article citing the software, is the appropriate granularity used? (RSMD-5.7)

#RSMD_cheklist

1. General Metadata Requirements

2. Accessibility & preservation

3. Reference & identification

4. Description & classification

5. Attribution & credit

6. Reuse, licensing & legal aspects

7. Re-execute: Dependencies & execution environment

Objective:

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

- ❑ Before choosing a license, did you verify who is the rights holder of the software? (RSMD-6.1)
- ❑ Is the license information available in the source code (intrinsic metadata file)? (RSMD-6.2)
 - ❑ If there are several licenses, are all defined in the software source code?
 - ❑ In the source code are the following attributes available alongside the license: name of the software, year, copyright holder, contact information (email), license identifier (e.g. SPDX) (RSMD-6.6)
- ❑ Is the license information available in the metadata record? (RSMD-6.3)
 - ❑ Is the license information in the code the same as the property on the metadata record?
- ❑ Are external software modules used identified with their authors and license? for compliance purposes, it is necessary to verify licenses compatibility (RSMD-6.4)
- ❑ Are the versions and contributions tracked? Using a VCS is preferable (RSMD-6.5)

#RSMD_cheklist

1. General Metadata Requirements

2. Accessibility & preservation

3. Reference & identification

4. Description & classification

5. Attribution & credit

6. Reuse, licensing & legal aspects

7. Re-execute: Dependencies & execution environment

Objective:

To ensure the usability of software and the ability to reproduce the same results in experiments, it is important that the software can be easily rebuilt and executed. This ensures that others can use the software effectively and achieve consistent outcomes.

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

#RSMD_cheklist

1. General Metadata Requirements

2. Accessibility & preservation

3. Reference & identification

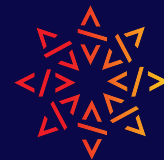
4. Description & classification

5. Attribution & credit

6. Reuse, licensing & legal aspects

7. Re-execute: Dependencies & execution environment

Making Metadata actionable and discoverable

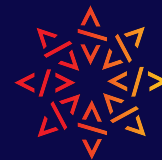


Research software integrates data, code, and metadata to support scientific understanding.



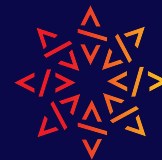
→ Metadata enables different actions:
discovery • citation • reproducibility • interoperability • recognition of research software

Goals



- ★ **Introduction:** what is at stake?
- ★ Building the software pillar of **Open Science**
- ★ Deep dive for **#BetterMetadata**
 - using the **RSMD Guidelines**
- ★ **Supporting research: new players in Academia**

Who are the stakeholders?



Researchers

- **archive and reference** software
- **find** useful software
- **get credit** for software
- **verify/reproduce/improve** results

Laboratories/teams

- **track** software contributions
- **produce** reports
- **maintain** web page



Research Organization

know its **software assets** for:

- technology transfer,
- impact metrics,
- Strategy

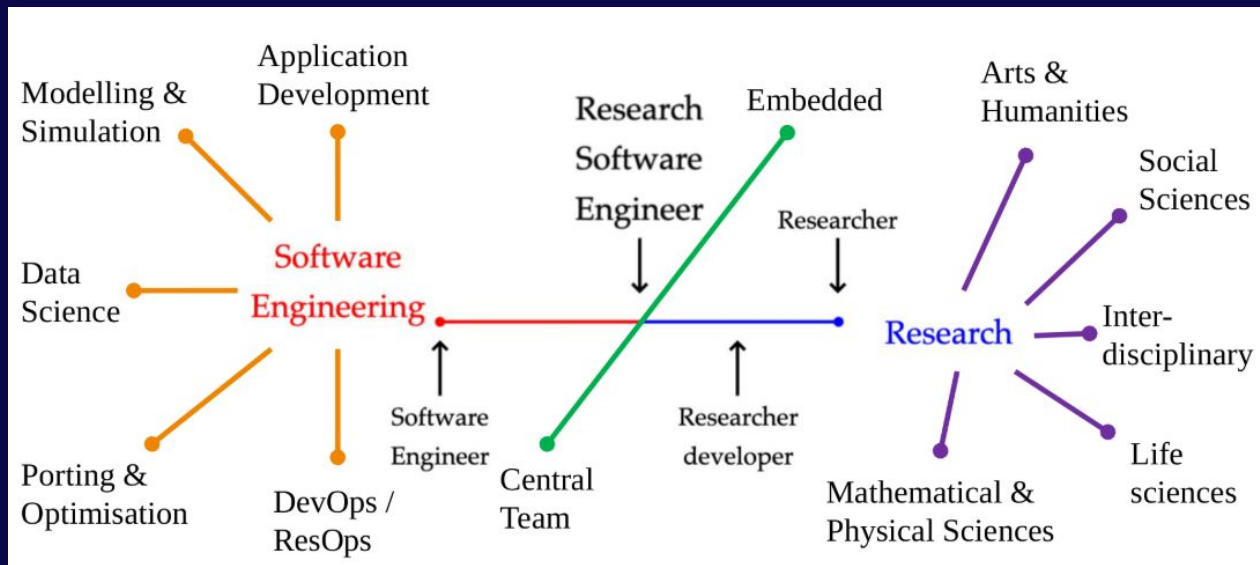
Curators

- **verify** and **curate** software metadata
- **provide** documentation on software curation
- **monitor** research teams' production

What about Research Software Engineers?

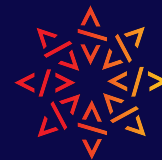
RSEs

- central role for creating and maintaining Research Software
- building bridges between Academia and Software Ecosystems
- support teams with technology



Chue Hong, Neil (2023). Is Research Software Engineering coming of age?. figshare. Presentation.
<https://doi.org/10.6084/m9.figshare.24078054.v5>

OSPPO for Open Source Programme Office

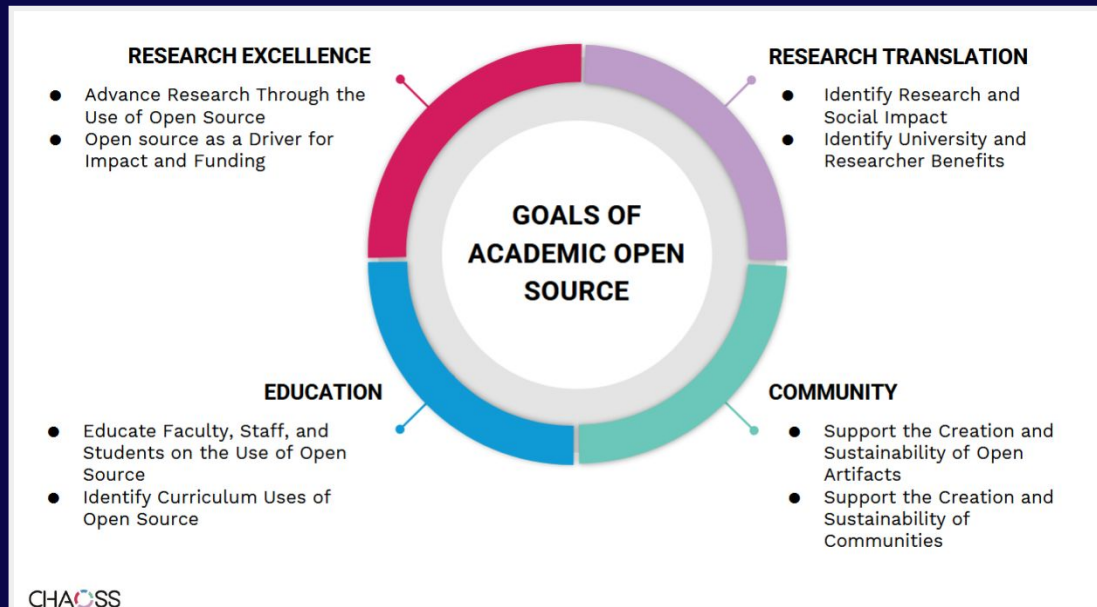


<https://code.gouv.fr/fr/blog/definition-ospo/>

An Open Source Program Office (OSPPO) is an entity within an institution that **defines and implements an open source strategy**.

OSPPO

- central role for Open Source strategy
- building bridges between Academia and Open Source
- support teams with technology



(C. Dillon, 2025)

[Personas full description](#)



Research libraries have a pivotal role

Software preservation requires a global and coordinated effort.

The long road ahead:

- availability,
- findability,
- traceability,
- reproducibility

Supporting the supporters: The **Archives and Libraries Interest Group (ALIG)**



Open infrastructure

Contribute to a 100% Open Source Infrastructure, valuing users input and support



Tailored Support

Receive assistance for your software archival questions, plans and projects



Community

Participate in an inclusive, innovation driven community

ALIG Members



Sant'Anna
School of Advanced Studies – Pisa



Canadian Research Knowledge Network
Réseau canadien de documentation pour la recherche



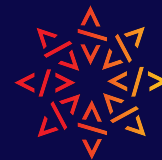
Konsortium der Schweizer Hochschulbibliotheken
Consortium des bibliothèques universitaires suisses
Consorzio delle biblioteche universitarie svizzere
Consortium of Swiss Academic Libraries

couperin.org
Consortium unifié des établissements universitaires et de recherche pour l'accès aux publications numériques



<https://www.softwareheritage.org/support/members-alig/>

The Software Metadata Curation Roadmap



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.
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.
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.

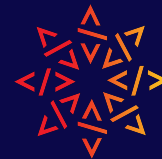


Read the full report

2024. Deliverable D6.1 | Report on
Standardisation and Curation of
Software Metadata and PIDs.
Zenodo.

<https://doi.org/10.5281/zenodo.14509418>

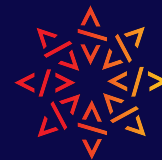
The Software Metadata Curation Roadmap



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.
3. **Automation:** Automation should be employed wherever possible to streamline the curation process and reduce the manual burden on researchers and curators.
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.

The Software Metadata Curation Roadmap



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.

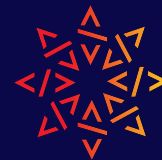


RSEs and **OSPOs** in Academia
can support **Librarians**

And connect *Academia* to
the rest of the *software world*



Call to action



Software recognition requires a **supported** and **coordinated** effort in **EOSC**

We need to treat code not just as a tool, but as a core pillar of our intellectual heritage. Metadata is part of this journey.



Adoption through community engagement:
What strategies do you believe are most effective
for building bridges between communities and
influencing adoption?





Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Thank you

morane@softwareheritage.org

<https://www.softwareheritage.org/>

Browse the **source code archive**

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

Become a **member**

<https://www.softwareheritage.org/support/members-align>

Volunteer as an **ambassador**

<https://www.softwareheritage.org/ambassadors/>

Subscribe to the **newsletter**

<https://www.softwareheritage.org/newsletter/>



The Team

© Inria / Photo B. Fourier



The Community

Chue Hong, N., Breitmoser, E., Antonioletti, M., Davidson, J., Garijo, D., Gonzalez-Beltran, A., Gruenpeter, M., Huber, R., Jonquet, C., Priddy, M., Shepeherdson, J., Verburg, M., & Wood, C. (2025). **D5.2 - Metrics for automated FAIR software assessment in a disciplinary context (1.1)**. Zenodo. <https://doi.org/10.5281/zenodo.15535629>

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. (2024). **D4.4 - Guidelines for recommended metadata standard for research software within EOSC (V1.0)**. Zenodo.

<https://doi.org/10.5281/zenodo.10786147>

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

Gruenpeter, M., Granger, S., Monteil, A., Sadowska, J., Nivault, E., Ioannidis, A., Azzouz-Thuderoz, M., Wagner, M., Bozaci, S., Wimalaratne, S., Bingert, S., & Cakir, G. (2024). **Deliverable D6.1 | Report on Standardisation and Curation of Software Metadata and PIDs**. Zenodo. <https://doi.org/10.5281/zenodo.14509418>

Ioannidis, A., Gruenpeter, M., Di Cosmo, R., Granger, S., Boyer, R., Douard, D., Lorentz, V., Lambert, A., Monteil, A., Sadowska, J., Nivault, E., Indarto, E., Steinhoff, W., Bozaci, S., Wagner, M., Didas, M., Azzouz-Thuderoz, M., Schubotz, M., Vergoulis, T., ... Gkinis, K. (2025). **Deliverable D6.2 - Report on Research Software Archival APIs and Connectors**. Zenodo. <https://doi.org/10.5281/zenodo.15552472>