Good practice versus reality: A landscape analysis of Research Software metadata adoption in European Open Science Clusters

1st Anas El Hounsri
Ontology Engineering Group
Universidad Politécnica de Madrid
Madrid, Spain
anas.elhounsri@alumnos.upm.es

2nd Daniel Garijo
Ontology Engineering Group
Universidad Politécnica de Madrid
Madrid, Spain
daniel.garijo@upm.es

***Abstract*—Research Software has become a key asset to support the results described in academic publications, enabling effective data analysis and reproducibility. In order to ensure adherence of Research Software to the Findable, Accessible, Interoperable, and Reusable (FAIR) principles, the scientific community has proposed metadata guidelines and best practices. However, it is unclear how these practices have been adopted so far. This paper examines how different scientific communities describe Research Software with metadata to support FAIR, how do they adopt existing good practices regarding citation, documentation or versioning, and what is the current adoption of archival services for long-term preservation. We carry out our analysis in the software registries of five science clusters (in domains ranging from Physics to Environmental Sciences), together with a multi-domain collaborative software registry. Our results highlight the main gaps in metadata adoption in the different communities, opening an opportunity for future contributions to aid researchers in adopting good FAIR and Open Science practices.**

***Index Terms*—Research Software, Metadata, FAIR software, FAIR principles, Guidelines.**

I. Introduction

Research Software, i.e., the code files, scripts, tools, or workflows involved in or produced throughout the research lifecycle [1] plays a pivotal role in modern scientific research, to support the research outputs described in scientific publications. Understanding the role of Research Software is essential not only for supporting research efficiency but also for ensuring reproducibility. In this regard, the scientific community has promoted initiatives towards adopting Open Science best practices, such as the Findable, Accessible, Interoperable and Reusable (FAIR) principles for data [2], which have also been extended for Research Software [3]. The adoption of FAIR is expected to ease the reuse of data and software while allowing automated systems to retrieve and interoperate with (meta)data. In fact, compliance with FAIR is now part of the agenda of international organizations such as the European Commission, through the European Open Science Cloud initiative [4], or NASA [5].[1]

In order to ease the adoption of FAIR and Open Science, several efforts have developed guidelines, lessons, and good practices for researchers to improve the metadata available in their code repositories (e.g., the Research Software MetaData Guidelines (RSMD) [6], Software Carpentry[2], Open Source Security Foundation[3], etc.). However, it is unclear how these practices are adopted by different scientific communities, making it challenging to assess their impact.

In this paper, we address this issue by assessing the adoption of Research Software metadata best practices (inspired by the RSMD guidelines) of a vast number of repositories from different multi-disciplinary scientific clusters. Our goal is to quantify in detail the adoption of Research Software metadata best practices within these communities and to find common gaps in their adoption. In particular, we focus on the following research questions:

- **RQ1:** How do communities describe Research Software metadata in their code repositories?
- **RQ2:** What is the adoption of archival infrastructures across disciplines?
- **RQ3:** How do software projects adopt versioning?
- **RQ4:** How comprehensive is the metadata provided in code repositories? Specifically:
  - What is the adoption of open licenses?
  - Do research projects include a description?
  - How well documented are research projects? (i.e., in terms of installation instructions, requirements, and documentation availability)
- **RQ5:** What are the most common citation practices among communities?

We carry out our analysis with code repositories available in public software registries of five major European Science Clusters, ranging from Physics to Environmental sciences, and a multi-disciplinary software registry. In total, our analysis includes 10,040 software entries.

[1] https://science.nasa.gov/open-science/

[2] https://software-carpentry.org/lessons/

[3] https://www.bestpractices.dev/en

With this work, our aim is to find the current adoption gaps in order to develop tools that will assist researchers in creating consistent software metadata records to support FAIR.

This paper is structured as follows. Section II refers to related work and studies in the area of FAIR principles and metadata adoption. In Section III, we introduce the clusters we chose to create our datasets. In Section IV, we explain our approach to collect data to answer these questions and methods of measurement. Next, in Section V, we provide descriptive statistics of the results. In Section VI, we present our analysis about each research question (RQ). We discuss the wider implications and the threats of not adopting FAIR. In Section VII we propose strategies to adopt best practices for the FAIR principles. We conclude the paper in Section VI.

II. Related Work

We divide this section into three main efforts: 1) existing guidelines for the adoption of FAIR in Research Software, 2) previous studies exploring metadata adoption by the scientific community, and 3) tools or managing Research Software metadata.

A. Guidelines for Open Science and FAIR Research Software

The FAIR principles, described by Wilkinson et al. (2016), [2] address key challenges in managing scholarly data by establishing guidelines to make data Findable, Accessible, Interoperable, and Reusable. These principles aim to support data findability and reuse not only by users but also by machines, which is essential in data-intensive research. The principles are intended for numerous stakeholders, including data publishers, software developers, funding bodies, and scientific communities.

FAIR has been adapted to different types of research artifacts. In Erdmann et al [7] the authors provide a structured introduction to FAIR concepts for different disciplines, one of them being Research Software, designed to help the research community implement the FAIR principles for both data and software. The FAIR for Research Software Principles (FAIR4RS) [3] followed, developed by an international working group, jointly led by the Research Data Alliance (RDA),[4] the Research Software Alliance (ReSA),[5] and FORCE11.[6] The FAIR4RS principles emphasize accommodating the unique characteristics of Research Software such as executability, versioning, and complex structures by focusing on specific challenges such as version identification, provenance tracking, and cross-environment usability. This work resulted in a detailed guide outlining principles for the entire lifecycle of Research Software, from creation to preservation and reuse. While other best practices for Research Software in support for Open Science have been developed throughout the years (e.g., Software Carpentry lessons, Open Source Security Foundation guidelines), FAIR Research Software has become a major goal for some major funding bodies (e.g., the European Commission [4]) and agencies (e.g., NASA[7]).

Adopting FAIR4RS, Chue Hong et al. [8] developed domain-agnostic metrics for the automated assessment of Research Software, addressing challenges such as software versioning, licensing, metadata management, and interoperability. These guidelines build on top of the Research Software MetaData recommendations (RSMD) [6], which indicate how Research Software metadata fields should be described in a given code repository, independently of its research discipline. RSMD guidelines inspire our work, which aims to determine current practices in different science domains.

Finally, other guidelines focus on different aspects of FAIR, such as citation and credit. Katz et al. [9] provide guidelines on software citation practices to ensure proper attribution and reuse within the academic publishing and research communities. Similarly, Bouquin et al. [10] identify software citation challenges, proposing interventions to support software acknowledgment across diverse research communities. Our RQ5 aims to explore the adoption of the citation practices suggested by these works.

B. Exploring Metadata adoption by scientific communities

Numerous studies have analyzed the various aspects of software metadata independently of their role in the FAIR principles. At a general level, Kelley and Garijo [11] create a Knowledge Graph of Research Software metadata with more than 10,000 Zenodo entries,[8] assessing the number of repositories with description, license or installation instructions among other metadata fields. Other works like [12] identify metadata standards that support reproducibility in computational research, from data input to final publication. By highlighting existing gaps and emerging trends, [12] offers targeted insights to improve reproducibility practices across all stages of scientific data analysis.

Other studies focus on the adoption of specific metadata fields. For example, software versioning is addressed by [13] with the goal of observing how Ansible role developers adhere to semantic versioning standards by analyzing changes across role versions, identifying common practices and inconsistencies, and developing a predictive model to help classify version changes. Meanwhile, [14] investigates whether software libraries on Maven Central adhere to semantic versioning principles, by examining the frequency and impact of breaking changes in major, minor and patch releases. Another example is [15], which aims to define the quality parameters for version control system (VCS) practices, develop a tool to automate the assessment of the quality of versioning activities based on VCS log files, and showcase its use in the evaluation of versioning practices in open source projects.

Several studies focus on licenses. Kapitsaki et al. [16] aim to ease the license compatibility process in open source software by using the Software Package Data Exchange (SPDX)

[4] https://www.rd-alliance.org/
[5] https://www.researchsoft.org/
[6] https://force11.org/
[7] https://software.nasa.gov/
[8] https://zenodo.org

standard to detect and address license compatibility issues in software packages. Zacchiroli et al. [17] compile a large-scale open data set of FOSS license texts and metadata from Software Heritage deposits in order to support historical studies in open source licensing.

As for the adoption of citation best practices, there are studies at different levels of granularity. Garijo et al. [18] explore how code repositories refer to their respective scientific publications in their code repositories, while Chaoqun Du et al. [19] focus on recognizing the tools mentioned in a research publication. Alternatively, A. Alsudais [20] investigates in-code citation practices in open Research Software libraries, specifically examining their prevalence, consistency, and completeness to propose best practices for citation standards.

Although these studies have investigated the adoption of specific elements in Research Software, such as licensing practices [16], versioning protocols [13, 14], citation [20] and general metadata [12, 11], the extent of metadata adoption that aligns specifically with the FAIR principles within scientific domains still remains largely unexplored. Studying this gap is highly relevant because the FAIR principles serve as a foundational framework for ensuring that Research Software and data are accessible, reusable, and interoperable within the Open Science community and identifying gaps and areas where metadata practices fall short of these principles is key to improve their adoption.

Finally, other quantitative works have focused on producing software metadata datasets in specific domains. For instance, in the Machine Learning domain, Jiang et. al. [21] create a dataset of pre-trained model metadata based on available model documentation (including nearly 30,000 GitHub repositories). Similarly, [22] develop a model metadata dataset from more than 15,000 entries in Hugging Face [23], Model Zoo [24] based on their documentation. While these quantitative studies may be used to assess the adoption of metadata in their respective domains, this was beyond the original scope of these works.

C. Tools for managing Research Software Metadata

Several tools exist to automate metadata extraction from software. For example, AIMMX [25] is a tool designed to automatically extract metadata from AI models on GitHub, in order to facilitate model management and reproducibility. The Software Metadata Extraction Framework (SoMEF) [26] is a tool for extracting Research Software metadata from code repositories like GitHub and GitLab, by exploring metadata files, README and public APIs. Instead, Codemetapy [27] and SOMESY [9] [28] are metadata harvesters based on package files (e.g., pom.xml for Java, setup.py for Python, etc.). These tools simplify the management of software project metadata by automating the synchronization of key metadata from the project. In our work, we adopt SoMEF to retrieve metadata to study the adoption of code repositories.

[9] https://materials-data-science-and-informatics.github.io/somesy/latest/

III. An overview of European Open Science Clusters

In order to assess software practices in different science communities, we focus our work in five of the six European Science clusters within the OSCARS network [29]. These science clusters represent major research communities in Europe, and all have at least a curated set of software repositories for their respective domains. We decided to leave out of our analysis the Social Sciences & Humanities Open Cloud (SSHOC), since it only had six repositories at the time of the analysis. An overview of the clusters is available below:

The European Science Cluster of Astronomy & Particle Physics ESFRI Research Infrastructures (ESCAPE)[10] is a collection of various research infrastructures in astronomy, particle physics, and nuclear physics to create solutions for managing large-scale scientific data. This work leverages open science principles and FAIR guidelines to improve data accessibility and usability across disciplines. Available in Zenodo, ESCAPE provides a repository of datasets, software, documentations, and papers that support scientific workflows and collaborative research. The repository includes resources for data processing, data preservation, and knowledge sharing, with the aim of creating a unified digital research infrastructure for the European science community.

The Environmental Research Infrastructure (ENVRI)[11] is a scientific hub which offers many notebooks to explore data and has a search service where users can find software such as notebooks and their corresponding GitHub repositories. ENVRI provides datasets, tools, and documentation that facilitate interdisciplinary research in Earth and environmental sciences. The repository aims to follow the FAIR principles, promoting shared access to data, models, and software relevant to climate change, biodiversity, and geosciences, it focuses on creating interoperable, reusable research tools, supporting scientists to easily integrate and share their work across disciplines contributing to environmental research.

The Life Science Research Infrastructure bio.tools (LS-RI)[12] is a directory dedicated to bio-informatics tools and resources that support life sciences research, facilitating data sharing, analysis, and software reuse. bio.tools provides metadata-driven descriptions for tools across bio-informatics, genomics, proteomics, and related fields. The project emphasizes metadata completeness, open access, and interoperability, supporting researchers in finding and using software solutions to analyze complex biological datasets. The repository's resources aid in standardizing software usage in life sciences, enhancing collaborative and reproducible research practices.

Photon and Neutron Open Science Cloud Software Catalog (PaNOSC)[13] project focuses on creating a software catalog for photon and neutron research facilities, providing accessible tools and services tailored to data-intensive scientific research.

[10] https://projectescape.eu/zenodo
[11] https://search.envri.eu/genericpages/genericpages?page=home
[12] https://bio.tools/bio.tools
[13] https://software.pan-data.eu/

By using a software database catalog that has all available software in the cluster, PaNOSC emphasize Open Science and FAIR principles to make software and data processing tools available for researchers. This repository offers tools for simulation, data reduction, and analysis, supporting scientists in areas such as materials science and structural biology. The catalog encourages standardization across facilities, promoting easier access and collaboration in photon and neutron sciences.

Finally, in our analysis, we also considered the *Research Software Directory (RSD)*.[14] While this software catalog does not represent a science cluster, it has gained momentum as a software registry from multiple science domains, with contributions from more than 20 different organizations. RSD is aimed at scholarly research software, in order to improve software visibility, citation, and reuse within the scientific community. RSD hosts metadata-rich entries that describe software functionality, authorship, and licenses, helping researchers find, evaluate, and incorporate software in their work.

IV. Methodology

Figure 1 shows an overview of the steps followed to perform the analysis of our five research questions. First, in order to extract the list of GitHub and GitLab repositories to analyze from each science cluster, we created a script that exploits domain registry APIs with web scraping, storing the list of repositories to analyze (with their respective source) in a JSON file[15][30]. The ESCAPE cluster provided their registry in Zenodo, allowing us to utilize Zenodo's API to retrieve metadata for each software entry, that includes links to repositories. The LS-RI cluster, specifically the bio.tools directory, serves as an aggregated directory of various sciences repositories and offer an API service, providing a structured access to metadata across its repositories. For the remaining resources, i.e., PaNOSC, ENVRI and RSD, did not offer a direct API nor standardized access to the repositories. Therefore we developed web scraping scripts to retrieve repositories from their sites [30]. It was necessary to create custom scripts to navigate each site structure, extract all accessible repository data, and organize it into a standardized format compatible with our dataset. Although web scraping caused additional challenges compared to previous clusters, this method allowed us to streamline the extraction of repositories from clusters that otherwise lack automated data retrieval solutions.

Next, we used SoMEF [26] on each code repository, in order to extract a structured and integrated metadata record. SoMEF extracts information from more than 30 metadata categories, indicating the source file for where each metadata property was found (e.g., README, LICENSE.md, etc.), which we use to answer some of the RQs. Additional scripts were developed[16] [31] in order to answer specific research questions. The dataset containing both the links to the repositories and the analysis with the final results is archived in Zenodo [32].

A. Research Questions Methodology

Our RQs are inspired by the RSMD guidelines, which divide RS metadata in five main categories **General, Accessibility and preservation, Reference and identification, Description, Credit and attribution**. In order to address each of our RQs, we searched for different files and metadata on each code repository, further detailed below:

To address **RQ1: how communities describe Research Software metadata?**, we explore the following metadata files:

- Citation File Format (CFF) [33]: Metadata file containing human and machine-readable citation metadata for research software. CFF is serialized in YAML, and includes fields for tracking license, title and url of a code deposit among others.
- README: Human-readable files, usually written in Markdown, with a brief description of the code repository or tool. READMEs usually contain a wide range of metadata attributes, including description, title, license, citation or installation instructions among other categories.
- Package files: Metadata files specific to programming languages (e.g., setup.py in Python, pom.xml in Java, etc.). These files are machine-readable.
- AUTHORS: A file for listing individuals or groups who contributed to the project, usually includes their names and optionally their roles
- CONTRIBUTORS: A file that details contributions of people involved in the project, often used with AUTHORS
- LICENSE: File specifying the license used in a code repository.
- codemeta.json: JSON-LD [34] file that contains structured metadata about the software, to facilitate discovery and citation.
- zenodo.json: JSON file with metadata to ease integration with Zenodo, specifying metadata for archiving software.

These categories address a range of metadata documentation types, from standardized files (Codemeta.json and Zenodo.json) that facilitate handling automated metadata, to human-generated files like README and CONTRIBUTORS that provide contextual and the individuals and/or group involved in their work. This selection is crucial to capture a comprehensive view of the ways in which researchers, software developers, and stakeholders maintain and share essential software metadata.

To examine **RQ2: adoption of archival infrastructures across different communities** we decided to explore whether code repositories are 1) deposited in Software Heritage,[17] which is the largest public archive of software source code and development history, with more than 16 billion unique source code files coming from more than 250 million collaborative development projects [35]; and 2) whether code

[14] https://research-software-directory.org/

[15] https://github.com/Anas-Elhounsri/Repositories-Extraction/tree/v0.1.0

[16] https://github.com/Anas-Elhounsri/Metadata-Adoption-Quantify

[17] https://www.softwareheritage.org/

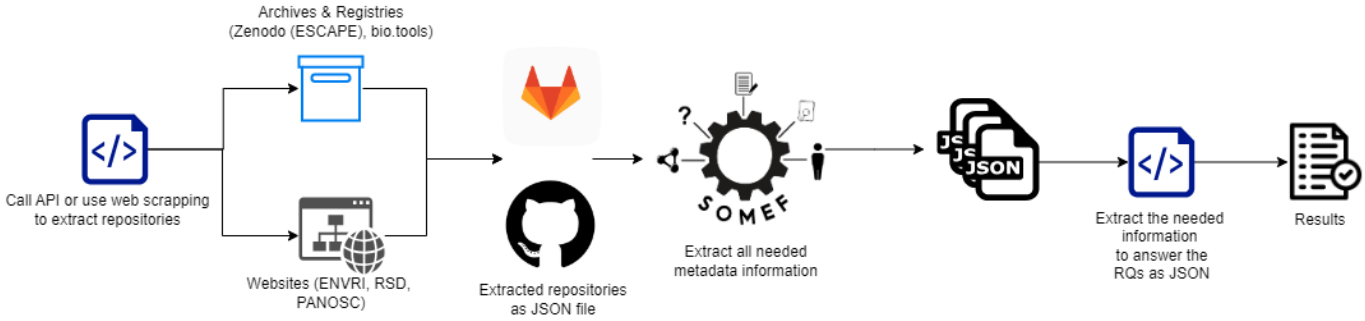

Fig. 1. Workflow to extract the information to answer the RQs

repositories acknowledge using Zenodo (i.e., including a badge in their README). Zenodo is a multidisciplinary open research repository that is developed and hosted by CERN [36]. We chose these infrastructures because they play a crucial role in ensuring that Research Software and outputs remain accessible, reproducible, and citable over time. By focusing on these archival solutions, we aim to assess how frequently they are adopted across various research fields. This approach underscores the importance of consistent use of archival platforms by researchers and software developers to ensure longevity and reach of their work.

For investigating **RQ3: how research software projects adopt versioning**, we observed whether software projects had code releases and consistent versioning naming scheme (i.e., all releases follow the same conventions) to provide a comprehensive understanding of versioning practices and how well are they documented. We look into the following versioning schemes:

- Semantic versioning[18]: the project uses releases following the "X.Y.Z" notation, where X stands for major changes, Y for minor changes and Z for patches.
- Calendar versioning: the project uses the current date to tag their releases
- Alphanumeric: the project combines numbers and characters in the release.

These elements are crucial as they ensure accurate identification and referencing of software projects, modules, and their respective versions. By examining the presence of releases, the consistency in versioning approaches and the adopted naming conventions, we aim to understand how projects manage their versions over time, which is crucial for ensuring clarity and reliability in RS and collaboration within the research community.

For the research question on **RQ4: how comprehensive is the metadata provided in code repositories?**, we selected attributes that represent critical aspects of metadata usage. We focus on three main attributes:

1) Project description: we assess if software projects have a "short description", (i.e., a one line summary of the project), or a longer description. The short description is provided by GitHub API, and the longer description is extracted with SoMEF from the README.md file. We aim to capture the depth and accessibility of project descriptions across platforms.
2) License: We assess whether 1) a license is declared; and 2) whether that license follows the SPDX[19] conventions (i.e., it is a known license). This allows us to quantify the degree to which communities follow recognized norms for software licensing, essential for compliance and interoperability.
3) Installation instructions: We assess whether installation instructions, covering installation steps, requirements, and documentation, are included in a code repository. We use SoMEF for the extraction, which looks into specific headers of the README files mentioning installation instructions, and links to external documentation (readthedocs,[20], wiki setup). In addition, we look for specific files describing requirements such as 'requirements.txt' in Python, 'pom.xml' in Java, etc.

Collectively, these attributes reveal how communities approach metadata and reflect their commitment to accessibility, traceability, and best practices in software metadata management.

For last research question **RQ5: what are the citation practices among the communities?**, we assess whether a citation is declared by any of the following methods:

- A BibTex [37] file (e.g., CITATION.bib) is available in the repository. BibTeX [37] is a popular reference management tool, used with LaTex to format bibliographies. ".bib" files are plain text files containing bibliographic information structured in a specific format.
- A Citation File Format file (CFF) is available in the repository. CFF files are a good practice that has recently seen uptake from the scientific community, receiving support from platforms such as GitHub[21]
- README contains a citation: alternatively, we assess if the README file contains a citation (i.e., by retrieving

[18] https://semver.org/

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

[20] https://docs.readthedocs.io/

[21] https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-citation-files

TABLE I
DISTRIBUTION OF REPOSITORIES OF DIFFERENT CLUSTERS

	ESCAPE	PANOSC	LS-RI	ENVRI	RSD
Number of Repositories	17	21	9058	512	432

BiBteX code in it), since many authors choose to suggest others how to cite their software directly with a BiBteX entry. The presence of citation details in README.md files represents a less formalized method of citation, but still widely used.

This methodology offers a flexible foundation that can be expanded to other software repositories, providing valuable insights into metadata adoption across diverse research communities. By leveraging APIs (when present) or using web scraping techniques, this approach can be applied to a variety of repository structures and metadata formats, making it possible to have a broader assessment of metadata completeness and FAIR compliance.

V. RESULTS

In this section we describe the results of our analysis. We first describe the distribution of the dataset, and then dive into each of the research questions. SoMEF was used mainly to extract the metadata from each code repository in the clusters. As discussed in Kelley and Garijo [11], SOMEF combines *synset analysis* (leveraging semantic relationships in natural language) with regular expressions and *machine learning classifiers* (for structured text parsing). This hybrid approach yields robust performance, with F1 scores spanning **0.82** for tasks like extracting long descriptions to **0.95** for metadata categories such as documentation. To further validate these results, the framework outputs were cross-checked against a subset of repositories, ensuring alignment with ground-truth metadata.

A. Dataset

As we mentioned in our methodology, ESCAPE and LS-RI offered public API services, while ENVRI, PaNOSC and RSD did not offer public APIs and therefore web scrapping techniques were used. In total 10,040 repositories were analyzed from all clusters. It is important to note that our dataset is highly imbalanced, with ESCAPE contributing 17 repositories, PaNOSC 21, ENVRI 512, RSD 432, and LS-RI contributing the majority with 9,058 repositories as shown in Table I. These reflect the current status of each catalog among the different science clusters.

B. Research Questions

The heat map in Figure 2 shows the main preferences of researchers when describing metadata. Note that the choices are not mutually exclusive, since a code repository may include several files for describing a license, citation or overall documentation. The “None” row indicates that none of the alternatives were chosen. Overall, most researchers add some kind of metadata to their source code repositories, but the practice seems to be inconsistent. When it comes to packages, we see somewhat a moderate adoption among communities except for ENVRI and LSRI. When it comes to having CONTRIBUTORS metadata, most communities are close to 0% except for ESCAPE with 17.6%. CFF adoption is also low, except in RSD. Codemeta.json and Zenodo.json do not have wide adoption (except for the ESCAPE community and RSD for Zenodo.json), indicating that while they are useful for interoperability of metadata records, authors still do not recognize them. However, we see the majority of communities provide README files with ESCAPE being 100%, showing the usefulness of these files.

TABLE II
RESULTS FOR RQ2: WHAT IS THE ADOPTION OF ARCHIVAL INFRASTRUCTURES CROSS THE DISCIPLINES?

Archival infrastructures	ESCAPE	PANOSC	LS-RI	ENVRI	RSD
Deposited in SWH	20%	82.61%	69.51%	85.33%	56%
Zenodo badge	41.18%	4.76%	4.5%	4.88%	28%

Table II shows the role of archival infrastructures across disciplines. For Software Heritage (SWH), the adoption rates differ significantly among communities, with ENVRI showing the highest adoption at 85.33%, followed by PANOSC at 82.61%, and a minimal adoption rate in ESCAPE at 20%. Repositories with a Zenodo badge exhibit a lower but relatively similar adoption overall, with ESCAPE leading at 41.18%, followed by RSD at 28% and LS-Ri, PANOSC and ENVRI have low adoption rates at 4.5%, 4.75% and 4.88%, respectively. It should be noted that having ”Zenodo Badge” does not necessarily mean that the repository is archived in Zenodo.

Next, Figure 3 and Figure 4 show how many projects have releases, along with the versioning scheme used, respectively. The results of the analysis of versioning practices in different research software projects provide insights into the adoption of versioning strategies. The data reveal that the percentage of projects with releases varies significantly, ranging from a low of 16% in the ENVRI project to a high of 88.2% in ESCAPE. In terms of a consistent version naming scheme, the findings indicate a similar trend, with ESCAPE 80% and ENVRI 80.5% code repositories employing a consistent naming scheme. LSRI, PANOSC and RSD show a decent adoption of consistent versioning schemes, although lower than ENVRI and ESCAPE. In general, these results highlight the varying levels of commitment to versioning practices across the projects analyzed. Some projects demonstrate a good approach to version management while others have low adoption of releases, but overall display a good versioning consistency. The ”Other” category in Figure 4 includes versioning schemes like ”version/4.7.1rc1” and ”i-vresse/wb-core@1.1.0” that do not classify one of the common versioning schemes.

Figures 5, 6 and 7 show the results for the adoption of specific metadata fields across the different science communities. When it comes to whether researchers provide a description of their code repositories, Figure 5 shows that most repositories

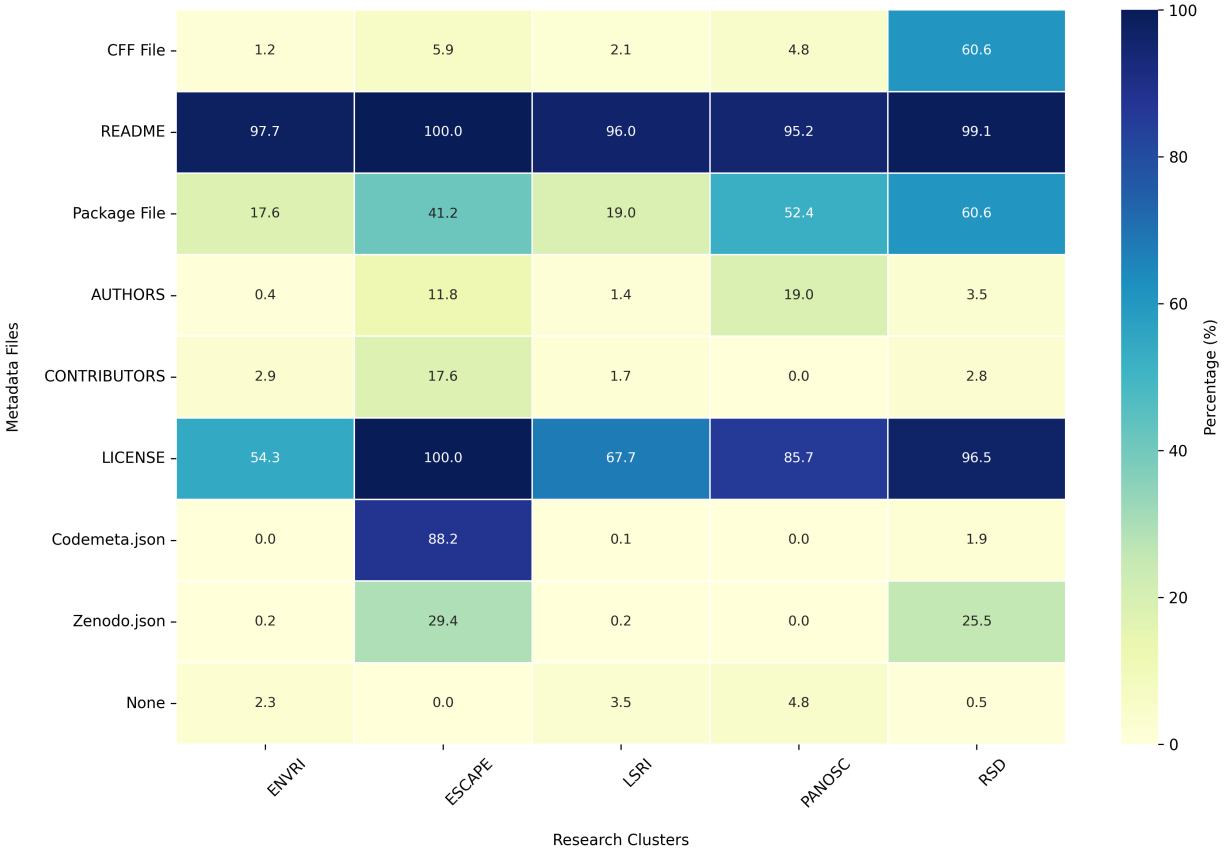

Fig. 2. RQ1: How do communities describe Research Software metadata in their code repositories?

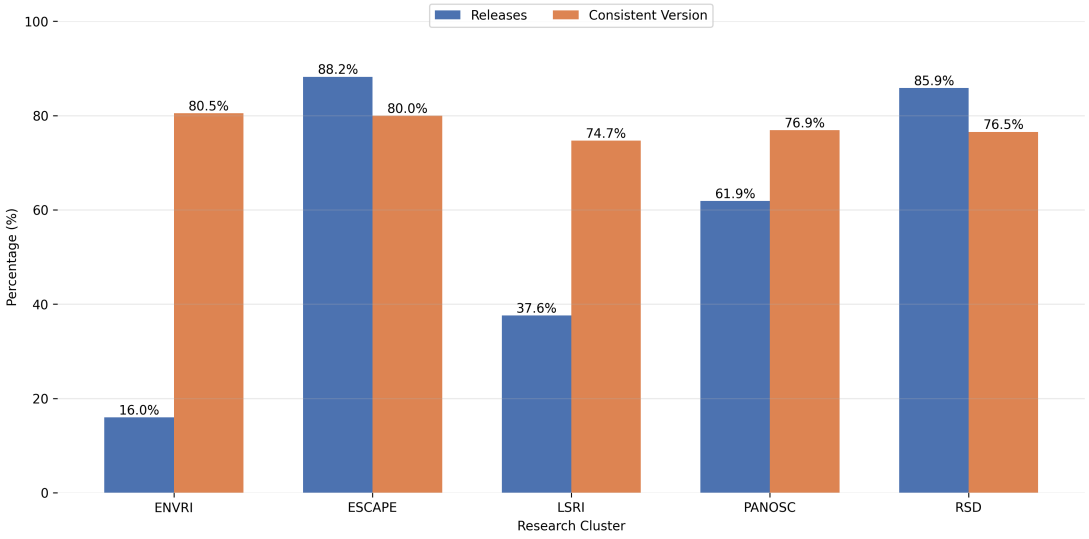

Fig. 3. RQ3-1: How do communities adopt releases and their consistency?

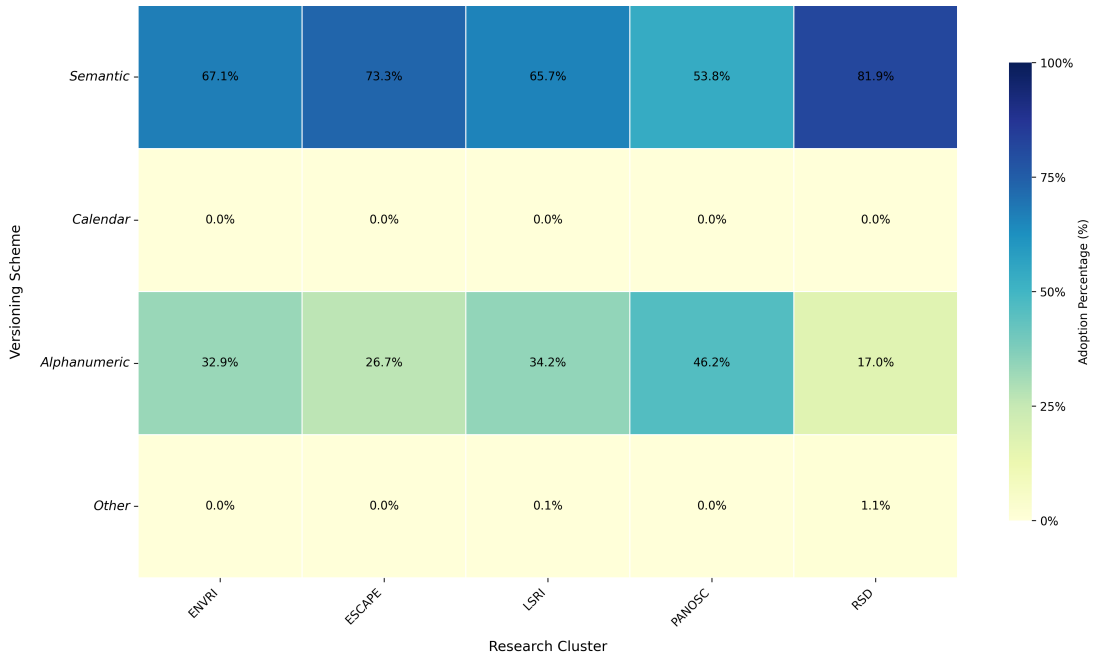

Fig. 4. RQ3-2: How do software projects adopt versioning?

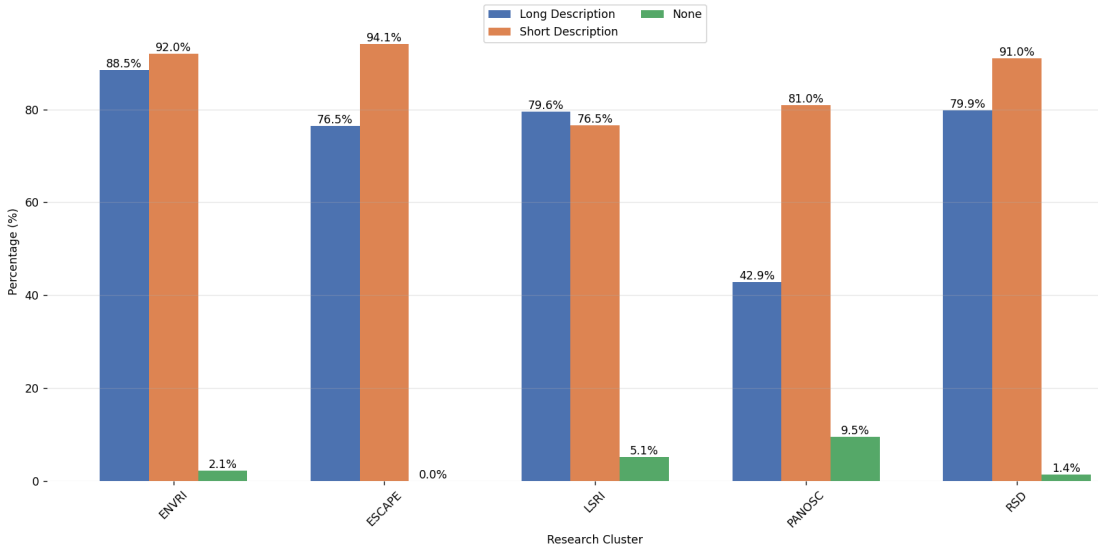

Fig. 5. RQ4-1: How do research projects include description?

across domains provide at least a short description of their intent. Long descriptions show between 5% (ENVRI) to 38% (PANOSC) less adoption than short descriptions, with LS-RI becoming an exception (nearly 4% of the projects have more long descriptions than short ones). We believe this could be the cuase since long descriptions are harder to develop. Next, Figure 6 shows the adoption of licenses. The majority of communities provide a known license by linking it to SPDX, except for ENVRI being the lowest with 49.3%, and also being the highest at not adopting any form of licenses at 45.35%. LSRI follows with 32.3%, while the other communities generally adopt a license in their repositories. Figure 7 shows the adoption of dedicated documentation besides a README file (e.g., through a readthedocs dedicated page, through wikis, etc.) and installation instructions (usually available in READMEs) or requirement files (e.g., requirements.txt, maven files, etc.). Here we have varying results, with installation having the highest adoption compared to other instructions, and requirements having the lowest adoption. PANOSC, LS-RI and RSD repositories tend to have specific sections for installation instructions (more than 50% for all three).

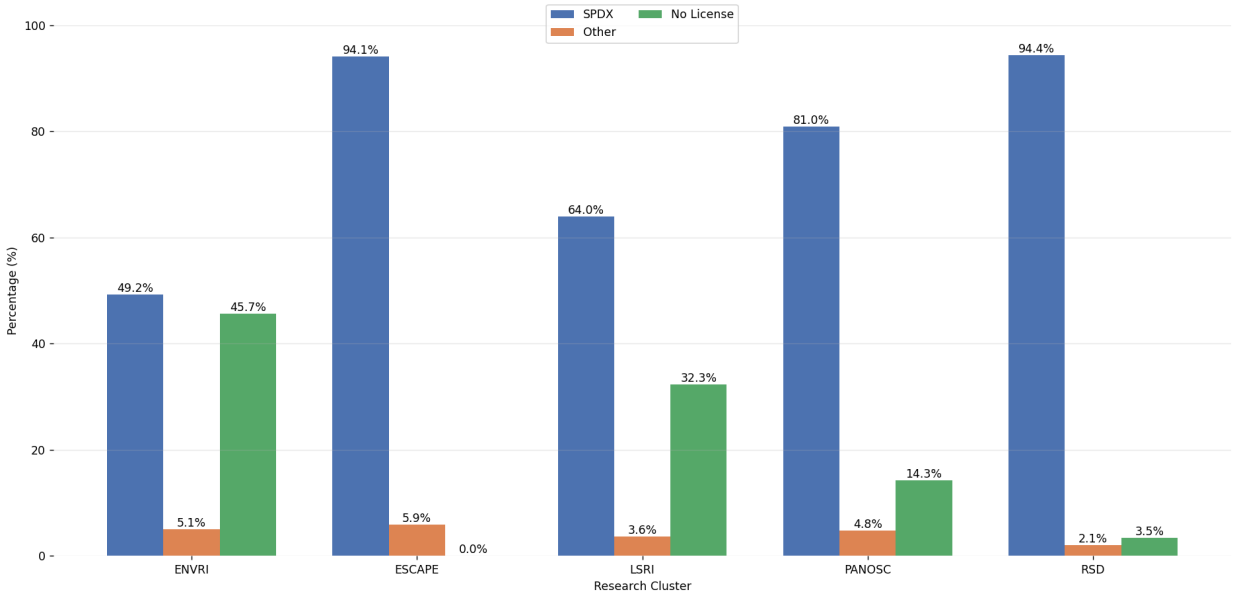

Fig. 6. RQ4-2: What is the adoption of open licenses?

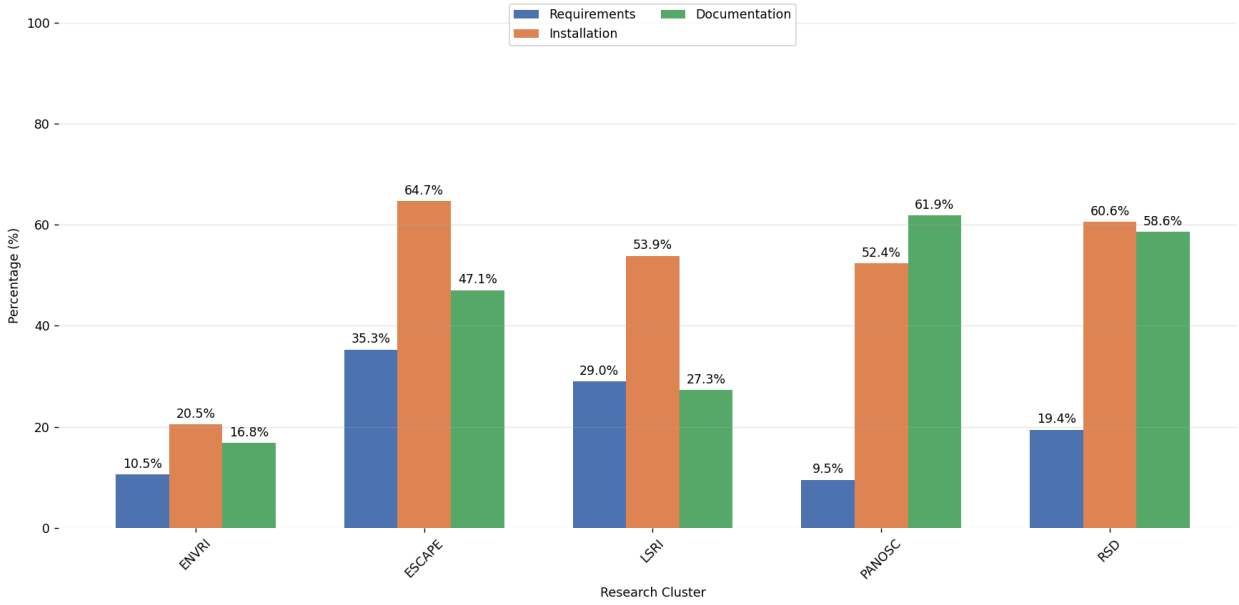

Fig. 7. RQ4-3: How well documented are research projects?

TABLE III
RESULTS FOR RQ5: WHAT ARE THE CITATION PRACTICES AMONG THE COMMUNITIES?

Citation Files	ESCAPE	PANOSC	LS-RI	ENVRI	RSD
.bib	5.88%	0%	4.37%	2.93%	2.55%
.cff	5.88%	4.7%	2%	1.17%	60.65%
README.md	17.65%	14.28%	31.03%	18.16%	5.79%
Citation	29.41%	18.89%	37.4%	22.26%	68.99%

Table III shows the results for common citation practices in the different communities. Note that these practices are also not mutually exclusive, since a repository may declare a CFF file while also providing a citation in the README file with the corresponding BibTeX code. A diversity of citation files are adopted, with .bib files being used in ESCAPE with 6%, ENVRI (3%) and LS-RI (4.3%), indicating a low level of structured BibTeX-based citation usage. In RSD BibTeX usage remains low at 2.5%, and PANOSC shows no adoption. The use of .cff files varies widely, RSD leads with a significant 60% adoption, where ESCAPE, PANOSC, LS-RI and ENVRI communities show minimal engagement at 6%, 4.7%, 2%, and 1%, respectively. README.md files appear more frequently as a

citation method across ESCAPE (17%), PANOSC (14%), LS-RI (312%) and ENVRI (18%). Notably, a varying proportion of projects in all science communities include citation (ranging from 22% in ENVRI to nearly 70% in RSD), indicating that a majority of Research Software projects across communities do not yet adopt citation practices (except for RSD).

VI. Discussion

In this section we address all research questions based on the result of our analysis. Our goal is to detect the main metadata practice adoption gaps in the different communities. We believe that while there is a dataset imbalance in the number of repositories (with little over 9,000 repositories for LS-RI) the division by community still uncovers unique trends (e.g., adoption of recent citation practices such as CFF).

The analysis for **RQ1: how communities describe RS metadata?** reveals that while researchers consistently dedicate effort in adding some metadata in their code repositories, the way they do so is inconsistent for the most part (LICENSE and READMEs are widely adopted, and package files are fairly commonly used, although ENVRI and LSRI have low adoption of packages). The adoption of recent community standards for metadata (CodeMeta) and citation (CFF) is still quite low, although they are present in some science communities (ESCAPE and RSD respectively). The continuous support from the community to help create these files (e.g., the CodeMeta generator,[22] cffinit[23]) may increase their adoption. Attribution-specific files like AUTHORS and CONTRIBUTORS are for the most part absent in code repositories, indicating that authors do not yet consider key to acknowledge contributor roles. The increasing demand for recognizing contributions (e.g., with the Credit taxonomy[24]) may see a rise in the adoption of these files. Zenodo.json files, used in the metadata integration with the platform are not widely recognized yet.

When it comes to the question **RQ2: What is the adoption of archival infrastructures across the disciplines?**, results show a varied commitment to long term preservation. Software Heritage (SWH) is notably used in ENVRI, PANOSC, LSRI and RSD while ESCAPE shows low adoption with 20%, suggesting potential reliance on alternative systems or low prioritization of preservation practices.

While many repositories use archival services, the results show that not many of them acknowledge their use. For example, ESCAPE software registry is in Zenodo, yet only 41.18% of their repositories include a badge with a DOI to the platform. RSD follows with (28%). The results highlight a significant variation in archival infrastructure adoption across disciplines, indicating that while some communities prioritize long-term software preservation, others may lack the necessary practices or awareness. As we can see that PANOSC, LS-RI and ENVRI dominate when it comes to archiving sources in SWH, but in Zenodo they have little use with most values being around 4%. Zenodo and Software Heritage are currently being integrated,[25] helping the automated propagation of metadata and records between both infrastructures.

Next, for **RQ3: how research software projects adopt versioning?** we see that practices in Research Software show major variations in the adoption of releases and consistent version naming schemes. While ESCAPE and RSD exhibit the highest adoption rates with 88.2% and 85.9% of projects having releases and 80% and 76.5% using consistent naming conventions, ENVRI shows much lower adoption, with only 16% including releases. A similar pattern applies to the rest of the communities. These results suggest that some communities prioritize versioning for clear identification and reliable collaboration, while others demonstrate less adherence to structured versioning practices. Overall, the results show us that there is a need for an easy way to keep consistent versioning. Here, an automated approach to guide researchers when creating new releases would improve versioning standards to ensure clarity and reliability in research software development.

Next, for **RQ4: How comprehensive is the metadata provided in code repositories?** we analyzed different metadata adoption for description, licenses and documentation associated with code repositories. For starters, by analyzing Figure 5, we see a positive adoption rate overall across all communities when it comes to including both short and long descriptions, which is probably due to how much easier it is to provide description with either a README file, or in the short description section in GitHub. When it comes to providing known licenses in Figure 6, we see a positive adoption rate, except for ENVRI, with an alarming number of repositories with no license. As for documentation and installation instructions in Figure 7, we notice a peculiar pattern in instructions, where only installation have the most positive adoption, with documentation slightly falling behind, and requirements show very low adoption rates, this may show a positive adoption of instructions if it was not for PANSOC and ENVRI being 9.5% and 10.5% in requirements, and ENVRI 16.8% in documentation. In this case, we believe there is an opportunity for registries to warn researchers when key metadata is missing from their repositories (e.g., license, which highly affects their reusability)

Finally, we used the results of Table III to answer **RQ5: what are the citation practices among the communities?**, seeing how different scientific communities adopt citations reveals several insights into how the communities approach this aspect in research software projects. The adoption of structured citation formats, such as .bib and .cff, is limited across most communities, with the only notable exception in RSD (60.6% for .cff). Unfortunately, not using standardized citation practices makes discoverability challenging. The frequent use of README.md files across communities, at least more notably than .bib and .cff (except for RSD with 5.78%) suggests that, despite the lack of structured formats, many projects still acknowledge the need for citation, providing this

[22] https://codemeta.github.io/codemeta-generator/
[23] https://citation-file-format.github.io/cff-initializer-javascript/
[24] https://credit.niso.org/
[25] https://blog.zenodo.org/2024/10/21/2024-10-21-swh/

information in a more accessible, though less formalized and standardized way. This gap of citation practices indicates a reliance on README files to communicate citation details more than using .bib and .cff, possibly due to their flexibility and ease of use, compared generating citation using .bib and .cff. However, while README files are easy to read, they lack a standard method of citing, which limits their effectiveness in automated citation systems. This gap is important for raising awareness in developing an easier and automated way to adopt novel citation formats such as CFF.

VII. Towards Best Practice Adoption

Improving the adoption of FAIR and Open Science practices in Research Software metadata requires a combination of strategies. The following recommendations cover a significant portion of good practices for the FAIR principles:

- **Create software metadata files:** Adding metadata files such as Codemeta and CFF enhances the findability and citation of Research Software. These files provide machine-readable metadata that aligns with the FAIR principles, enabling automated systems to retrieve key information about a software tool. Tools like the CodeMeta Generator[26] simplify the creation of these files, making it easier for researchers to adopt them. By including metadata files, researchers ensure that their software is better integrated into the scholarly ecosystem and can be cited appropriately.
- **Improve the structure of your README file:** A well-structured README file serves as the primary entry point for good understanding of a software repository. By following existing guidelines,[27] researchers can include essential details such as project descriptions, installation instructions, usage examples, and contact information. This not only improves the usability of the software tool, but also reduces the barrier for others to adopt and contribute to the project. A comprehensive README ensures clarity and promotes better collaboration within the research community.
- **Reuse existing Licenses:** Licensing is a critical aspect of Research Software that determines how it can be used, modified, and shared. By adopting existing licenses like SPDX, researchers ensure compliance with legal requirements and promote reusability across different software projects. Standardized licenses simplify the process of understanding usage rights and reduce the risk of incompatibilities when integrating software from multiple sources. This practice also ensures trust and transparency within the research community.
- **Use consistent versioning in software releases:** Implementing a consistent versioning scheme, such as semantic versioning, alphanumeric, calendar etc. ensures that changes to the software are clearly communicated. This practice provides users with a reliable way to track updates, identify compatibility issues, and understand the scope of changes in each release. Consistent versioning enhances the reliability and maintainability of Research Software, making it easier for users to adopt and integrate into their workflows.
- **Archive your software releases:** Archival platforms like Zenodo and Software Heritage play a crucial role in preserving Research Software for the long term. By depositing software in these repositories, researchers ensure that their work remains accessible, citable, and reproducible. Archiving also protects against data loss and provides a permanent record of the software's development history. The GitHub-Zenodo bridge[28] simplifies archiving by allowing researchers to directly link GitHub repositories to Zenodo, automatically generating DOIs for software versions.

VIII. Conclusions and Future Work

This paper provides a landscape analysis of metadata adoption practices across five major European Science Clusters. Inspired by the RSMD guidelines [6] and leveraging an automated workflow, we examined common practices to address key research questions about metadata documentation, archival service usage, code versioning, repository descriptions, and citation behaviors. Our findings highlight gaps in current practices and serve as a foundational step toward understanding barriers to metadata standardization in scientific communities, while raising awareness about the importance of adopting robust practices.

In future work, we aim to broaden this analysis beyond European clusters to include domain-specific registries such as ASCL (astrophysics), SWMath (mathematics), and CSDMS (hydrology). By expanding to these communities, we will assess how automated tools and policies influence the incremental adoption of metadata practices over time. Additionally, we plan to refine automated detection methods to identify recurring pitfalls and generate tailored recommendations for researchers, empowering them to implement the FAIR4RS principles more effectively. This dual focus on scalable analysis and actionable guidance will close the gap between policy frameworks and practical adoption, emphasizing on a culture of reproducibility and transparency in Open Science.

References

[1] Morane Gruenpeter et al. *Defining Research Software: a controversial discussion*. Version 1. Dec. 2021. DOI: 10.5281/zenodo.5504016.

[2] Mark D. Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific Data* 3 (Mar. 2016), p. 160018. DOI: 10.1038/sdata.2016.18.

[26] https://codemeta.github.io/codemeta-generator/

[27] e.g., https://everse.software/RSQKit/how_to_make_a_good_readme

[28] https://docs.github.com/en/repositories/archiving-a-github-repository/referencing-and-citing-content

[3] Neil P. Chue Hong et al. *FAIR Principles for Research Software (FAIR4RS Principles)*. Version 1.0. May 2022. DOI: 10.15497/RDA00068.

[4] European Commission, Directorate-General for Research, and Innovation. *Strategic Research and Innovation Agenda (SRIA) of the European Open Science Cloud (EOSC)*. Publications Office of the European Union, 2022. DOI: doi/10.2777/935288.

[5] Daniel C. Berrios, Afshin Beheshti, and Sylvain V. Costes. "FAIRness and Usability for Open-access Omics Data Systems". In: *NASA Technical Reports* (2018).

[6] Morane Gruenpeter et al. *D4.4 - Guidelines for recommended metadata standard for research software within EOSC*. June 2023. DOI: 10.5281/zenodo.8097537.

[7] Christopher Erdmann et al. *Top 10 FAIR Data & Software Things*. Feb. 2019. DOI: 10.5281/zenodo.2555498.

[8] Neil Chue Hong et al. *D5.2 - Metrics for automated FAIR software assessment in a disciplinary context*. Version 1.0 - DRAFT not yet approved by the European Commission. Oct. 2023. DOI: 10.5281/zenodo.10047401.

[9] Daniel S. Katz, Neil P. Chue Hong, Tim Clark, et al. "Recognizing the value of software: a software citation guide [version 2; peer review: 2 approved]". In: *F1000Research* 9 (2021), p. 1257. DOI: 10.12688/f1000research.26932.2.

[10] Daina Bouquin et al. *Advancing Software Citation Implementation (Software Citation Workshop 2022)*. 2023. arXiv: 2302.07500 [cs.DL].

[11] Aidan Kelley and Daniel Garijo. "A framework for creating knowledge graphs of scientific software metadata". In: *Quantitative Science Studies* 2.4 (Dec. 2021), pp. 1423–1446. ISSN: 2641-3337. DOI: 10.1162/qss_a_00167.

[12] J. Leipzig et al. "The role of metadata in reproducible computational research". In: *Patterns* 2.9 (2021), p. 100322. DOI: 10.1016/j.patter.2021.100322.

[13] Ruben Opdebeeck et al. "Does Infrastructure as Code Adhere to Semantic Versioning? An Analysis of Ansible Role Evolution". In: *2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. 2020, pp. 238–248. DOI: 10.1109/SCAM51674.2020.00032.

[14] Steven Raemaekers, Arie van Deursen, and Joost Visser. "Semantic Versioning versus Breaking Changes: A Study of the Maven Repository". In: *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*. 2014, pp. 215–224. DOI: 10.1109/SCAM.2014.30.

[15] Rickard Elsen, Inggriani Liem, and Saiful Akbar. "Software versioning quality parameters: Automated assessment tools based on the parameters". In: *2016 International Conference on Data and Software Engineering (ICoDSE)*. 2016, pp. 1–6. DOI: 10.1109/ICODSE.2016.7936139.

[16] G. M. Kapitsaki, F. Kramer, and N. D. Tselikas. "Automating the license compatibility process in open source software with SPDX". In: *Journal of Systems and Software* 131 (2017), pp. 386–401. DOI: 10.1016/j.jss.2016.06.064.

[17] Stefano Zacchiroli. "A large-scale dataset of (open source) license text variants". In: *Proceedings of the 19th International Conference on Mining Software Repositories*. MSR '22. ACM, May 2022. DOI: 10.1145/3524842.3528491.

[18] Daniel Garijo et al. "Bidirectional Paper-Repository Tracing in Software Engineering". In: *21st International Conference on Mining Software Repositories*. Cham: ACM, 2024. DOI: 10.1145/3643991.3644876.

[19] Chaoqun Du et al. "Softcite dataset: A dataset of software mentions in biomedical and economic research publications". In: *Journal of the Association for Information Science and Technology* 72.7 (2021), pp. 870–884. DOI: 10.1002/asi.24454.

[20] A. Alsudais. "In-code citation practices in open research software libraries". In: *Journal of Informetrics* 15.1 (2021), p. 101139. DOI: 10.1016/j.joi.2021.101139.

[21] Wenxin Jiang et al. "PeaTMOSS: A Dataset and Initial Analysis of Pre-Trained Models in Open-Source Software". In: *Proceedings of the 21st International Conference on Mining Software Repositories*. MSR '24. Lisbon, Portugal: Association for Computing Machinery, 2024, pp. 431–443. ISBN: 9798400705878. DOI: 10.1145/3643991.3644907.

[22] Wenxin Jiang et al. " PTMTorrent: A Dataset for Mining Open-source Pre-trained Model Packages ". In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2023, pp. 57–61. DOI: 10.1109/MSR59073.2023.00021.

[23] Hugging Face. *Hugging Face – The AI Community Building the Future*. https://huggingface.co/. [Online; accessed 2024]. 2024. URL: https://huggingface.co/.

[24] ONNX. *ONNX Model Zoo*. https://github.com/onnx/models. [Online; accessed 2022]. 2024. URL: https://github.com/onnx/models.

[25] Jason Tsay et al. "AIMMX: Artificial Intelligence Model Metadata Extractor". In: *Proceedings of the 17th International Conference on Mining Software Repositories*. MSR '20. Seoul, Republic of Korea: Association for Computing Machinery, 2020, pp. 81–92. ISBN: 9781450375177. DOI: 10.1145/3379597.3387448.

[26] Allen Mao, Daniel Garijo, and Shobeir Fakhraei. "SoMEF: A Framework for Capturing Scientific Software Metadata from its Documentation". In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 3032–3037. DOI: 10.1109/BigData47090.2019.9006447.

[27] Maarten van Gompel. *CodeMetaPy*. Version 2.5.3. Version 2.5.3. Amsterdam, Netherlands: KNAW Humani-

ties Cluster, 2024. URL: https://github.com/proycon/codemetapy.git.

[28] Mustafa Soylu et al. *somesy*. Version 0.4.3. 2024. URL: https://materials-data-science-and-informatics.github.io/somesy.

[29] OSCAR Project. *OSCAR Science Clusters*. https://oscars-project.eu/science-clusters. [Online; accessed 2024]. URL: https://oscars-project.eu/science-clusters.

[30] Anas El Hounsri and Daniel Garijo. *Anas-Elhounsri/Repositories-Extraction: Initial version*. Version v0.1.0. Feb. 2025. DOI: 10.5281/zenodo.14803009.

[31] Anas El Hounsri and Daniel Garijo. *Anas-Elhounsri/Metadata-Adoption-Quantify: Full release*. Version v1.0.0. Feb. 2025. DOI: 10.5281/zenodo.14803019.

[32] Anas El Hounsri and Daniel Garijo. *Dataset for "Good practice versus reality: A landscape analysis of Research Software metadata adoption in European Open Science Clusters"*. Zenodo, Feb. 2025. DOI: 10.5281/zenodo.14770578.

[33] Stephan Druskat et al. *Citation File Format*. Version 1.2.0. Aug. 2021. DOI: 10.5281/zenodo.5171937.

[34] JSON-LD Community Group. *JSON-LD: JavaScript Object Notation for Linked Data*. https://json-ld.org/. [Online; accessed 2024]. URL: https://json-ld.org/.

[35] Roberto Di Cosmo and Stefano Zacchiroli. "The Software Heritage Open Science Ecosystem". In: *Software Ecosystems*. Springer International Publishing, 2023, pp. 33–61. ISBN: 9783031360602. DOI: 10.1007/978-3-031-36060-2_2.

[36] CERN. *Zenodo*. 2024. URL: https://openscience.cern/zenodo.

[37] Graduate Academic Affairs. *Creating Bibliography with LaTeX*. 2024. URL: https://webmaster.iit.edu/files/graduate-academic-affairs/latex-bibliography-help.pdf.