



European
Commission

Scholarly Infrastructures for Research Software

Report from the
EOSC Executive
Board Working
Group (WG)
Architecture Task
Force (TF) SIRS

Independent
Expert
Report

EOSC Executive Board
WG Sustainability
December 2020

Research and
Innovation

Scholarly Infrastructures for Research Software

European Commission
Directorate-General for Research and Innovation
Directorate G — Research and Innovation Outreach
Unit G.4 — Open Science
Contact Corina Pascu
Email Corina.PASCU@ec.europa.eu
RTD-EOSC@ec.europa.eu
RTD-PUBLICATIONS@ec.europa.eu
European Commission
B-1049 Brussels

Manuscript completed in December 2020.

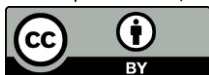
The European Commission is not liable for any consequence stemming from the reuse of this publication.
The views expressed in this publication are the sole responsibility of the author and do not necessarily reflect the views of the European Commission.

More information on the European Union is available on the internet (<http://europa.eu>).

PDF	ISBN 978-92-76-25568-0	doi: 10.2777/28598	KI-02-20-998-EN-N
-----	------------------------	--------------------	-------------------

Luxembourg: Publications Office of the European Union, 2020

© European Union, 2020



The reuse policy of European Commission documents is implemented based on Commission Decision 2011/833/EU of 12 December 2011 on the reuse of Commission documents (OJ L 330, 14.12.2011, p. 39). Except otherwise noted, the reuse of this document is authorised under a Creative Commons Attribution 4.0 International (CC-BY 4.0) licence (<https://creativecommons.org/licenses/by/4.0/>). This means that reuse is allowed provided appropriate credit is given and any changes are indicated.

For any use or reproduction of elements that are not owned by the European Union, permission may need to be sought directly from the respective rightholders.

Cover page: © Lonely #46246900, ag visuell #16440826, Sean Gladwell #6018533, LwRedStorm #3348265, 2011; kras99 #43746830, 2012. Source: Fotolia.com.

Scholarly Infrastructures for Research Software

***Report from the EOSC Executive Board Working
Group (WG) Architecture Task Force (TF) SIRS***

Edited by: the EOSC Executive Board

December 2020



Table of Contents

1	EXECUTIVE SUMMARY	5
2	INTRODUCTION	7
2.1	Scope and Goals	9
2.1.1	Archive, Reference, Describe, Credit: The Four Pillars	10
2.2	Infrastructures Participating in the TF	11
2.2.1	Archives	12
2.2.2	Publishers	16
2.2.3	Aggregators	20
3	STATE OF THE ART	24
3.1	Survey on Related Initiatives and Related Works.....	24
3.1.1	Archives	25
3.1.2	Publishers	27
3.1.3	Aggregators	28
3.2	Summary of the State of the Art Presentations in the Group.....	29
3.2.1	Archives	30
3.2.2	Publishers	30
3.2.3	Aggregators	31
3.3	Best Practices and Open Problems.....	32
3.3.1	Best Practice Principles for Archives	32
3.3.2	Best Practice Principles for Publishers	34
3.3.3	Best Practice Principles for Aggregators	35
3.4	Cross-cutting Concerns	36
3.4.1	Metadata	36
3.4.2	Identifiers	37
3.4.3	Quality and Curation	37
3.4.4	Metrics	38
3.4.5	Guidelines	38
3.4.6	Tools and Workflows	39
4	THE ROAD AHEAD	40
4.1	General Requirements	41
4.1.1	Archive	41
4.1.2	Reference	42
4.1.3	Describe	44
4.1.4	Cite/Credit	45
4.1.5	Easing Adoption	46
4.2	Exemplarity Criteria for Participating Infrastructures.....	47
4.2.1	Accommodating Innovation	48
4.3	Possible Workflows	48
4.3.1	Self-Archiving	49
4.3.2	Scholarly Publication with Associated Source Code	53
4.3.3	Aggregators	60
5	RECOMMENDATIONS	62
5.1	Funding Development of Tools, Standards, and Guidelines	62
5.1.1	Interactions	62
5.1.2	Metadata About Software	63
5.1.3	Identifiers	63
5.1.4	Credit	63
5.1.5	Policy/Guidelines	64

5.1.6	Easing Adoption	64
5.2	Broader Policy Recommendations for the EOSC.....	65
5.2.1	Criteria of Excellence, and Sustainability of the Architecture	65
5.3	Longer Term Perspectives	66
5.3.1	Advanced Technology Development	66
5.3.2	Policy	67
6	ANNEXES	68
6.1	Glossary	68
6.2	Bibliography	70
6.3	Task Force Participants	76
6.3.1	Roberto Di Cosmo (Chair TF SIRS)	76
6.3.2	Jose Benito Gonzalez Lopez (Co-Chair TF SIRS)	77
6.3.3	Jean-François Abramatic (Chair WG Architecture)	78
6.3.4	Kay Graf	79
6.3.5	Miguel Colom	79
6.3.6	Paolo Manghi	80
6.3.7	Melissa Harrison	81
6.3.8	Yannick Barborini	82
6.3.9	Ville Tenhunen	83
6.3.10	Michael Wagner	83
6.3.11	Wolfgang Dalitz	84
6.3.12	Jason Maassen	85
6.3.13	Carlos Martinez-Ortiz	85
6.3.14	Elisabetta Ronchieri	86
6.3.15	Sam Yates	87
6.3.16	Moritz Schubotz	87
6.3.17	Leonardo Candela	88
6.3.18	Martin Fenner	89
6.3.19	Eric Jeangirard	90

FOREWORD

Software has become another medium for people to share knowledge. In the case of research, software delivers knowledge using programming languages the same way publications deliver knowledge using natural languages.

In the 21st century, many research activities use computing systems to monitor their experiments, to visualise or analyse their results, or to check hypotheses through simulation.

It has therefore become essential to archive, preserve and share research software. Pioneering efforts have started to ensure persistence and availability of software next to publications and data.

In order to fulfil its promise to lead Europe towards Open Science, EOSC, the European Open Science Cloud programme, has to support the development of Scholarly Infrastructures for Research Software. In June 2020, the EOSC Architecture Working Group (WG) launched a Task Force (TF) with the mandate to offer recommendations on this topic. With the partnership of nine organizations involved in research software infrastructures, the TF was able, in four months, to produce this report.

After providing an initial view of the state of the art of research software infrastructures, the report suggests best practices, identifies open problems and describes use cases. On this basis, recommendations are proposed both at the technical and policy levels with immediate as well long-term horizons.

As the chair of the EOSC Architecture WG, I want to warmly thank the TF members for the depth of the study and the look into the future that the report delivers.

Jean-François Abramatic

October 16, 2020

1 EXECUTIVE SUMMARY

The TF on Scholarly Infrastructures of Research Software, as part of the Architecture WG of the European Open Science Cloud (EOSC) Executive Board, has established a set of recommendations to allow EOSC to include software, next to other research outputs like publications and data, in the realm of its research artifacts. This work is built upon a survey and documentation of a representative panel of current operational infrastructures across Europe, comparing their scopes and approaches.

This report summarizes the state of the art, identifies best practices, as well as open problems, and paves the way for federating the different approaches in view of supporting the software pillar of EOSC.

As the fuel of innovation, the engine of our industries and a fundamental pillar of academic research, software is a necessary component of modern scholarly research. Hence, software developments emerge across many fields and disciplines. Unfortunately, often forgotten is the important fact that software is actually a special form of knowledge, designed by humans to be read by humans, executed by machines, in the form of *software source code*. Software source code allows the description of data visualisation, data analysis, data transformation, and data processing in general with a level of precision that goes way beyond what can be achieved in scholarly articles. It is now well recognized that without access to the software used in research projects, it is extremely difficult to reproduce scientific results, and to build upon the results obtained by other researchers.

Over the past decade, awareness has been raised about the importance of software in the scholarly world. Several infrastructures have started to be built, or adapted, to address some of the following key challenges that need to be tackled to put software on equal footing with other research outputs in the scholarly world:

1. *Archiving* software to ensure research software artifacts are not lost.
2. *Referencing* software to ensure research artifacts can be precisely identified.
3. *Describing* software to easily discover and identify research software artifacts.
4. *Crediting* all authors to ensure their contributions are recognized.

To start addressing these challenges, the TF was formed by representatives of the EOSC Architecture WG together with representatives from current operational infrastructures across Europe (presented in Section 2.2 Infrastructures Participating in the TF). The TF covers the full spectrum of *archives*, *publishers*, and *aggregators* (including catalogues) and is considered a representative panel based on their wide-ranging experience in addressing some of the challenges involved in building the four pillars.

The TF considers that addressing these needs will require *establishing standards*, *developing tools*, *improving and interconnecting infrastructures*, *training*, *outreach*, and *involvement with the publishing community*. *Proper funding* will need to be provided both for the *development*, *communication*, and *outreach efforts*, and for the *operational costs*.

The TF concretely delivered a set of recommendations that emerged from the analysis of the current needs and state of the art, and the design of the future architecture. They include short term actionable items, broader policy recommendations for the EOSC, as well as a longer-term perspective.

Short term recommendations are foreseen to be turned into concrete development projects in a 2–4-year time-frame. The concrete recommendations detailed at the end of this report have the objective to (i) *strengthen interactions* between archives, publishers, and

aggregators, (ii) adopt *metadata standards*, (iii) *generalize the use of extrinsic and intrinsic identifiers for software*, (iv) *ensure appropriate citations* for research software source code, (v) foster standardization through *policy* and *guidelines*, and (vi) *ease adoption* of the processes and tools for the research community at large.

The TF foresees that the EOSC has a key role to play in ensuring the overall architecture will be built in a way to best cater to the needs of the research community. To ensure openness, transparency, and good governance, the EOSC should elaborate a set of criteria of excellence, incorporating these principles, for its participating infrastructures, and should provide concrete recommendations. Additionally, the EOSC should actively get involved with the key infrastructures for software, take part in their strategic evolution and earmark proper funding to ensure their long-term sustainability.

The longer-term perspectives include objectives that should be taken up in the roadmap to be addressed over a 4–7-year horizon. Of importance is the development of advanced technology, such as open plagiarism detection technology and advanced search engines for software source code. Moreover, technology and tools should be explored to address a proper integration between different research outputs: articles, data, and software.

Lastly, the TF strongly recommends including a clause in all future research funding programs to request research software is made available under an Open Source license by default, and that all deviations from this default should be duly motivated. While EOSC subscribes to the general statement that all research output should be "*as open as possible, as closed as necessary*" it is believed that stimulating this default is needed for software to be put on equal footing with other research outputs.

The consultation period ran from October 21 until November 10. All comments received were considered.

2 INTRODUCTION

Software has become a fundamental component in the modern scholarly ecosystem and software developments emerge across all fields and disciplines (Van Noorden et al., 2014). It is now well recognized that without access to the software used in research projects, data is less suitable for reuse (Baker, 2016) and confirmation (Barnes, 2010; Stodden et al., 2012).

In the scholarly world, software has been often seen just as a tool, overlooking the important fact that software is actually a special form of knowledge, written by humans for humans, in the form of software source code, and only later turned into executable code for a machine (Abelson & Sussman, 1985; Shustek, 2006).

Software source code implements and describes data generation and collection, data visualisation, data analysis, data transformation, and data processing with a level of precision that is not met by scholarly articles alone. Publicly accessible software source code allows a better understanding of the process that leads to research results, and open source software allows researchers to build upon the results obtained by others, provided proper mechanisms are put in place to make sure that software source code is preserved and that it is referenced in a persistent way (Di Cosmo, Gruenpeter, & Zacchiroli, 2020).

Researchers have always written research articles to present their results. The growing trend, however, lies in the fact that they more and more include software to support or demonstrate such results. This latter activity can represent a significant part of their work and must be properly taken into account when researchers are evaluated by their peers and institutional authorities (Alliez et al., 2020; Clément-Fontaine et al., 2019).

Last, but not least, software source code developed by researchers is only a thin layer on top of the complex web of software components, most of them developed outside of academia, that are necessary to execute the software and produce scientifically meaningful results (K. Hinsén, 2019): as an example, Figure 1 shows the broad sets of software components that are needed to use the popular matplotlib library (Hunter, 2007).

As a consequence, scholarly infrastructures that support software source code written in academia must go the extra mile to ensure they adopt standards and provide mechanisms that are compatible with the ones used by tens of millions of non-academic software developers worldwide.

And yet, much is left to be done when it comes to providing adequate support for ensuring that the source code of software related to research activities is preserved for the long term, properly identified and described, with credit given to those that contribute to it (Meeting, 2019).

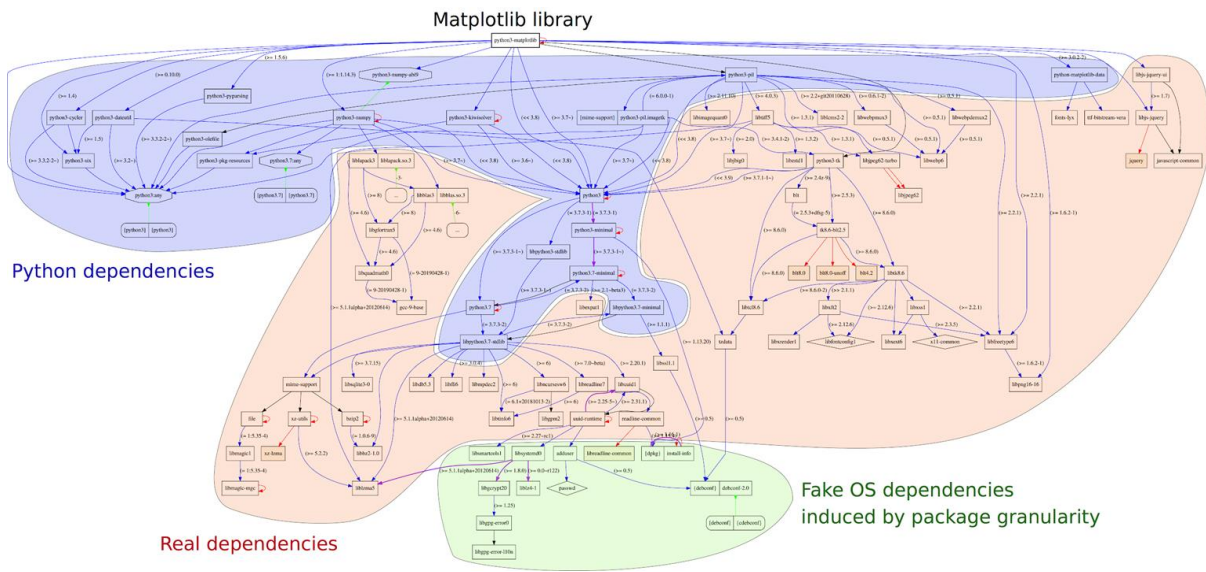


Figure 1. Example of the complexity in direct and indirect dependencies for a specific python package (matplotlib). Boxes represent actual packages (libraries that need to be installed on the system), arrows indicate dependencies to other packages, labels indicate the minimal/maximal version number. In blue the Python dependencies, in red the “true” system dependencies incurred by python (e.g., the libc or libjpeg62), in green some “fake” dependencies incurred by the package management system but which are very likely not used by python (e.g., adduser or dpkg).

For decades, we have seen software source code made available through development platforms that are not meant to be archives, and referenced in research articles using links to them, or just mentioned with their name (Howison & Bullard, 2015). We all knew that links to these platforms may rot (Spinellis, 2003), and that the platforms themselves may go away, but only recently the extent of the danger has started to be appreciated, with the closing down of huge platforms like Google Code¹ and Gitorious.org², and the phasing out of support for the Mercurial version control system (VCS) in Bitbucket³.

As a result, millions of software projects have been displaced or lost, and the web of scholarly knowledge has been significantly endangered.

Here are a few out of many examples that show how this phenomenon manifests itself:

- The link to the source code on the web page of the WorldView project from MIT⁴ now points to a long-gone repository on Gitorious.org: <http://www.gitorious.org/worldview>.
- Authors of articles that thought their source code was safe, discover a few years later that it is lost: see for example what is reported in (Di Cosmo & Danelutto, 2020) or this recent tweet.

1 <https://www.softwareheritage.org/2016/09/01/google-code-content-now-safely-collected-in-software-heritage/>

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

3 <https://www.softwareheritage.org/2020/04/23/rescuing-250000-endangered-mercurial-repositories/>

4 <https://projects.csail.mit.edu/worldview/about/>



Gabriel Altay
@gabrielaltay

Just realized [@Bitbucket](#) disabled all mercurial repositories when the [@asclnet](#) informed me that a link associated with an old paper of mine was down. Thought all was lost, but someone archived all the repos! very classy move by [@octobus_net](#) and [@SWHeritage](#).

[Traduire le Tweet](#)

1:48 AM · 31 août 2020 · Twitter Web App

In the research world, some initiatives started to arise in order to address this issue of paramount importance overall but particularly for open science. Scholarly repositories like Zenodo (in 2014), provided a mechanism for researchers to self-archive their research software, manually, or automatically directly from GitHub.

Software Heritage (Abramatic et al., 2018; Di Cosmo & Zacchiroli, 2017) is taking over the heavy lifting of proactively harvesting and archiving all software source code with its full development history (including, luckily, all the examples above). It is important that all scholarly repositories, which may be of varying sizes and addressing different institutional or disciplinary needs, properly interface with Software Heritage and offer researchers the additional functionalities they expect, and that research articles reference the archived version of the software.

Indeed, even though the source code in the examples above is now preserved, the links in research articles that point to the original development platform are now broken. The source code is less likely to be found, reused, or able to demonstrate its support for the research findings, and the research articles have also lost some of their content (the source code). So, although software archival is absolutely necessary, it is also a necessity to reference the archived version in research articles in a way that ensures the research remains supported by the code and the reuse potential is maximised, generalising, in particular, the use of intrinsic identifiers.

Archiving and referencing archived versions within research content is just the beginning. We need proper identifiers for the artefacts and for the metadata, to deal with tens of available ontologies for describing software, ensure all required metadata is easily available for citation purposes, and fill the huge gap that we face when looking for support for software credit and citation (referencing tools and publisher house styles).

Addressing these needs will require *standards, tools, infrastructures, training, outreach, and involvement with the publishing community*. It also needs *proper funding* both for the *development, communication, and outreach efforts*, and for the *operational costs*.

2.1 Scope and Goals

The Scholarly Infrastructures of Research Software (SIRS) TF was assembled with the clear mission to explore current practices and approaches, identify best practices and open problems, and formulate concrete recommendations for *a global architecture of infrastructures* that will allow EOSC to put *software source code* on a par with articles and data.

As clearly stated in (Clément-Fontaine et al., 2019), “*software is a hybrid object in the world of research as it is equally a driving force (as a tool), a result (as proof of the existence of a solution) and an object of study (as an artefact)*. This specific status means we need to define strategies, tools and procedures which are adapted to the various issues it raises. These include the citation of contributions to software design and production, the

reproducibility of research results involving software and the wider usage and long-term sustainability of the software heritage created."

Due to this polymorphic nature of software in the world of research, the term "*research software*" may carry very different meanings in different research communities: in this report, we will use this term simply to designate *software that researchers in any discipline may feel the need to have scholarly infrastructure support for*, no matter if it is considered a tool, a result or an object of study⁵.

From the outset it is important to clarify that we are well aware of the many difficult challenges that need to be tackled when one tries to ensure that a given executable or a full software system can be reliably run again, enabling full reproducibility of research results, as well as of the complex organizational, economic, and strategy issues that need to be addressed for its sustainability.

The focus of the work of this TF is different, as we have on purpose addressed only *software source code in the world of research*, for two main reasons. First and foremost, the source code of software is *human readable knowledge*, and embodies precious technical and scientific information that cannot be extracted from the executables, and that can be understood even when the corresponding executable can no longer be run. Second, properly addressing the issues that handling software source code raises for scholarly infrastructures is a significant challenge by itself, as will be clearly outlined in this report, and it is easier to provide actionable recommendations by focusing on this first.

2.1.1 *Archive, Reference, Describe, Credit: The Four Pillars*

As we have seen above, *software source code* in the research world is quite different from research data for a number of reasons (Katz et al., 2016), including two particularly important ones. First, software is an executable tool, with complex execution semantics that make each piece of software a node in an intricate dependency network. Second, software source code is authored by humans as part of doing research, whereas most research data represents recorded observations.

Hence it is not surprising that the popular FAIR (Findability, Accessibility, Interoperability, and Reuse) Guiding Principles for research data (Wilkinson et al., 2016) do not fit it well, as they were not designed for it. It is not our purpose to discuss how FAIR principles should be modified, or even entirely overhauled, to be of use when dealing with software: other working groups are grappling with this challenge⁶.

We focus here on *four key concrete issues* that need to be tackled to put software on equal footing with other research outputs, and where scholarly infrastructures play a prominent role:

- **Archiving** software that has been developed up until now to ensure research software artifacts are not lost (Abramatic et al., 2018);
- **Referencing** software to ensure research software artifacts can be precisely identified (Di Cosmo, Gruenpeter, & Zacchiroli, 2020);

⁵ An important remark is that *the very same software* may be at the same time a tool for some researcher, a result of the research of another, and the object of study of a third one.

⁶ See for example the FAIR4RS Working Group of the Research Data Alliance, FORCE11, the Research Software Alliance, FAIRsFAIR task 2.4 on "FAIR services and software" (Gruenpeter et al., 2020) and (Lamprecht et al., 2020): <https://www.rd-alliance.org/groups/fair-4-research-software-fair4rs-wg>

- **Describing** software to easily discover and identify research software artifacts (Bönisch et al., 2012; Di Cosmo, Gruenpeter, & Zacchiroli, 2020);
- **Crediting** to ensure proper credit is given to authors (Alliez et al., 2020).

We believe that a global architecture of scholarly infrastructures must provide appropriate means to Archive, Reference, Describe, and Credit software source code in the world of research, that we refer to as the four pillars, also abbreviated as ARDC, in the following.

To this end, we brought together a representative panel of current operational infrastructures across Europe that deal with software source code written by researchers, covering the full spectrum of archives, publishers, and aggregators (including catalogs).

Over the four months of collaborative work, these representatives helped develop a deep understanding of the issues at stake, based on their concrete and factual experience in addressing some of the challenges involved in building these pillars.

2.2 Infrastructures Participating in the TF

The SIRS TF consists of representatives of the EOSC Architecture WG and representatives of *operational infrastructures that deal with software source code written by researchers*. The following section is dedicated to the introduction of the participating European infrastructures. These infrastructures are classified into three groups, depending on the *primary goal* of the infrastructure: *Archives*, *Publishers*, and *Aggregators*.

In the context of this report we use the term *Archives* to designate services that have as one of their primary goals the *long-term preservation* of the digital content that they collect. This includes a broad spectrum of services, ranging from *institutional repositories*⁷ to *disciplinary repositories*⁸ in the scholarly world, as well as services that have a broader scope.

Publishers are organizations that prepare submitted research texts, possibly with associated source code and data, to produce a publication and manage the *dissemination*, *promotion*, and *archival process*. Software and data can be part of the main publication, or assets given as *supplementary materials* depending on the policy of the journal. In addition, publishers implement a process for ensuring the quality of the accepted research material (usually peer review), which is carried out by a subject-specific community of experts.

Finally, we use the term *Aggregators* to designate services that collect information about digital content *from a variety of sources* with the primary goal of increasing its *discoverability*, and possibly adding value to this information via processes like curation, abstraction, classification, and linking. These services, that include scholarly catalogues and indexes, usually provide a search engine that gives access to a description of the aggregated content, and may provide links to versions of it archived elsewhere. These services may be generalistic, or have a disciplinary, geographic, or institutional scope.

The following summary sheets present the nine infrastructures that are represented in the SIRS TF: three for the *archives* category (HAL, Software Heritage, and Zenodo), three for the *publisher* category (Dagstuhl Publishing, eLife, and IPOL), and *three* in the aggregators category (OpenAIRE, ScanR, and swMath).

⁷ See https://en.wikipedia.org/wiki/Institutional_repository

⁸ See https://en.wikipedia.org/wiki/Disciplinary_repository

2.2.1 Archives

The following list provides the set of archives that are represented in the SIRS TF:

- HAL
- Software Heritage
- Zenodo

These infrastructures are a respectable representation of the Archive subgroup landscape, as the services vary greatly in geographic scopes (national and international); scope in terms of content (universal and scholarly), size, number of registered users, number of software projects handled, and their typical workflow.



Headquartered in **Lyon, France**



Category **Archives**

HAL

French multidisciplinary repository

Type	Year of creation
Public	2001
Legal status	(opened to software in 2018)
Service in organisation	
Geographical scope	
National	
Content scope	
Academic	

Software projects handled	
Source code archival	610
Third party: Software Heritage	

Supported identifiers	
Extrinsic identifiers for software metadata, Intrinsic identifiers (SWHID) for software source code	
Curation	
Metadata: manual curation Source code: no	

Estimated resources	
Software infrastructure	FTE:20
Licence: open source Data access: open API	

Policy support (optional)
Institutional and national endorsement

Website
hal.archives-ouvertes.fr

Reference Publications
Curated Archiving of Research Software Artifacts: Lessons Learned from the French Open Archive (HAL) Journal Article International Journal of Digital Curation, 15 (1), pp. 16, 2020. <https://doi.org/10.2218/ijdc.v15i1.698>

Organization Overview

Created in 2001, HAL became the national multidisciplinary open archives platform chosen by French universities and research institutions in 2013. This enables researchers to deposit their scientific outputs to ensure their preservation and availability for the world scientific community, and for society in general (visit the website hal.archives-ouvertes.fr). HAL is operated by the Center for Direct Scientific Communication (CCSD), a joint service unit whose supervisory authorities are the CNRS, Inria, INRAE, and the University of Lyon, with financial backing from the Ministry of Higher Education, Research and Innovation (MESRI). Today, HAL has around 20 FTE working on the open archives platform. In 2018 a collaboration between CCSD, Inria, and Software Heritage resulted in the opening of a new scientific deposit in the national open archive which started to accept software specific items. Currently, more than 600 software items have been archived in HAL.

Scope

The geographic scope of HAL is mainly national. However, the archive is open to the international public and users as well. HAL has no scope restrictions in terms of content. Deposits are accepted from all scientific fields. HAL addresses a local user base. It has a database with more than 230,000 registered users.

Workflow

The author deposits the software (compressed archive) in HAL and completes the associated metadata to ensure an accurate description is provided. Some metadata, such as affiliations, authors, disciplines, journals and funders is linked to authority files. The author can also link the software to the publication persistent identifier (PID, eg DOI) upon filing.

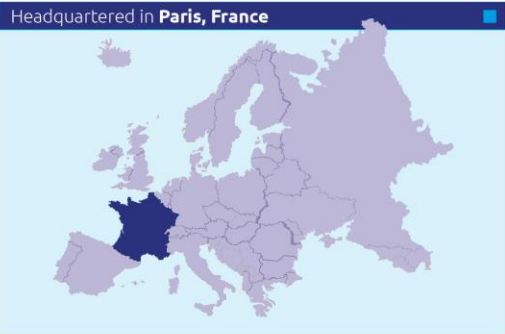
To ensure data quality, software deposits are moderated and curated before being put online.

Once the deposit is validated, software is pushed to the Software Heritage archive. Referencing is performed using automatic linking with Software Heritage PIDs.





Software Heritage



Category Archives

Software Heritage

Universal software archive

Type Private not for profit	Year of creation 2015
Legal status Hosted organisation	
Geographical scope International	
Content scope Global	

Software projects handled
140 Million

Source code archival Own storage plus mirror network	
Supported identifiers Intrinsic	
Curation Full traceability of archived source code and metadata. Curation & review are out of scope.	

Estimated resources
Current: 14 FTE, 1,600,000€/year
Target: 50 FTE, 10,000,000€/year

Software infrastructure Licence: open source Data access: open API	
Policy support (optional) National Plan For Open Science (France) Agreement with UNESCO	
Website softwareheritage.org	
Demo Video https://www.youtube.com/watch?v=8nISvYh7Vpl	
Reference Publications (Di Cosmo, 2020) doi:10.1007/978-3-030-52200-1_36 (Abramatic et al., 2018b) doi:10.1145/3183558	

Organization Overview

Software Heritage is a nonprofit foundation hosted by the Inria foundation that builds and maintains an universal source code archive. The organization was created in 2015 and is headquartered in Paris, France. Today, Software Heritage has around 14 FTE working on the project and has already archived over 9 billion source code files and 140 million software projects from all over the world.

Scope

Software Heritage has an international scope and is universal in the sense that it archives all software, academic and non-academic. Software Heritage addresses a variety of needs, from cultural heritage, to industry, to research, for all software developers, users, and researchers worldwide.

Workflow

The source code can be archived in three ways. First, Software Heritage archives source code that is regularly and proactively harvested from development and distribution platforms worldwide. Second, archival of particular software origins can be triggered manually or programmatically by any user, via a "save code now" interface. Third, a deposit API is made available to identified partners, such as HAL and IPOL, that can push content to the archive. Specific data structures are used for archival to ensure deduplication and provide cryptographically strong intrinsic identifiers. These identifiers have a standardized schema, called SWHID, registered with IANA and currently adopted in the SPDX industry standard as well as in Wikidata. In the context of scholarly infrastructures for research software, archival and reference are the core of Software Heritage's mission. However, the organization also contributes actively to ongoing efforts to standardize metadata for source code description. It is also involved in efforts to simplify software citation: for example, the biblatex-software package recently released allows cite software with proper links in the Software Heritage archive.





Headquartered in **CERN Geneva, Switzerland**



Category **Archives**

Zenodo

Global multidisciplinary repository

Type Public	Year of creation 2013
Legal status Service in organisation	
Geographical scope International	
Content scope Global	

Software projects handled 44,086 (more than 101K versions)	
Source code archival Own storage	
Supported identifiers Extrinsic (intrinsic supported only as relations metadata)	
Curation Metadata: no Source code: no	

Estimated resources FTE:4	
Software infrastructure Licence: open source (MIT) Data access: open API	
Policy support (optional) International endorsement	

Website
zenodo.org

Organization Overview

Zenodo was founded in 2013 by CERN under the umbrella of the OpenAIRE for the European Commission, with the aim to develop a catch-all repository (visit the website zenodo.org). Today, Zenodo is run by CERN and OpenAIRE remains the main partner, but Zenodo now receives funding from other organizations as well to work on specific projects. It is a free service that is headquartered at CERN, Geneva, Switzerland. All sorts of research-related objects can be archived in Zenodo, including publications, documentations, data, and software. However, today, Zenodo is the provider of more than 80% of all software DOIs worldwide. Around 44,000 software items have been handled and 101,000 software items including versions. The Zenodo team consists of four persons on average.

Scope

The geographic scope of Zenodo is international. Moreover, the scope in terms of content is not restricted to specific scientific domains. Zenodo addresses all researchers in general, and in particular the long-tail of science, and has a current user base of 101,538 active users.

Workflow

The main infrastructure has two entry points; a user interface (UI), which can be used by anyone to submit information, and APIs that can be used by anyone to programmatically submit content to Zenodo. Software developers can, besides using the UI and APIs, automatically submit content to Zenodo through GitHub. Another component of Zenodo, which is part of a project to solve the challenge around software citation within the scientific field of astronomy, is the citations broker. This is a microservice that is gathering PID-relations information (e.g. PID1 is the same as PID2, or PID1 cites PID2) from many different discovery services, disambiguates software IDs and stores these relations in a database in such a way that Zenodo, or any other service, can query and display relevant software citation information. Through the APIs and UI, users or programs can get the DOIs assigned to the records, all the metadata, usage statistics, versioning, and citations. This information can also be fed back into GitHub to use it for citation purposes.



2.2.2 *Publishers*

The following list provides the set of publishers that are represented in the SIRS TF:

- Dagstuhl Publishing
- eLife
- IPOL

These infrastructures are a respectable representation of the Publisher subgroup landscape, as the services vary greatly in size, number of registered users, number of software projects handled, and workflow.



SCHLOSS DAGSTUHL
Leibniz-Zentrum für Informatik

Headquartered in **Wadern, Germany**



Category **Publishers**

Dagstuhl

Open access computer science publisher

Type

Private not for profit

Year of creation

1990

Legal status

Independent organisation

(Dagstuhl Publishing in 2008)

Geographical scope

International

Content scope

Disciplinary

Software projects handled

100

Source code archival

Own storage

Supported identifiers

Extrinsic

Curation

Metadata: manual curation
Source code: high level of review
(by Artifact Evaluation Committees)

Estimated resources

FTE: 41
(Publishing 3 FTE)
Annual budget:
~ 3 Million EUR
Complete
Dagstuhl incl.
Seminar Center

Software infrastructure

Licence: partly closed
and partly open source
Data access: open API



Website
dagstuhl.de



Reference Publications

Wagner, Michael. (2017, May). Hitting the Bull's Eye with DARTS: Artifact Evaluation in Computer Science. Zenodo.

<http://doi.org/10.5281/zenodo.583007>

Organization Overview

Schloss Dagstuhl is an open access editorial infrastructure that belongs to the Leibniz Center for Informatics (visit the website dagstuhl.de). It was originally founded in 1990 as a seminar center and is headquartered in Wadern, Saarland, Germany. Besides their main activity as seminar host, Dagstuhl operates and maintains the dblp computer science bibliography and the open access publishing department, Dagstuhl Publishing (founded in 2008). The publishing unit has 3 FTE employees. In their Dagstuhl Artifacts Series (DARTS), Dagstuhl publishes evaluated research data and artifacts in all areas of computer science. In total, around 100 software artifacts, including data, software, and other tools, have been handled. Dagstuhl's most important series is LIPIcs (Leibniz International Proceedings in Informatics), which is a series of high-quality conference proceedings across all fields in computer science. Last year, just over 1200 articles of 29 conferences were published. In these series Dagstuhl has most experience in referencing software and research artifacts in general.

Scope

Dagstuhl has an international scope. The scope in terms of content is disciplinary in the sense that mostly Computer Science and Informatics articles are published through Dagstuhl. Moreover, the scope is limited to conference proceedings, which matches the publication culture of the targeted discipline.

Workflow

Dagstuhl focusses on artifacts supplementing the textual contribution instead of explicit support for software in general. Submitted manuscripts are scanned for references to software artifacts supplementing the text. Artifacts published by Dagstuhl have their own metadata and DOI. The metadata is collected via a submission system and from artifact descriptions. The artifacts are published separately from the related paper, with their own metadata and so potentially have different authors. All artifacts published in Dagstuhl's journal DARTS pass an "artifact evaluation" before the journal is published.



Headquartered in **Cambridge, United Kingdom**



Category **Publishers**

eLife

Open research communication in biology & medicine

Type

Private not for profit

Legal status

Independent organisation

Geographical scope

International

Content scope

Disciplinary
Life sciences and medicine

Year of creation

2012

Software projects handled

819

Source code archival

Third party: GitHub

Supported identifiers

Extrinsic

Curation

Metadata: yes
Source code: no

Estimated resources

FTE:50

Software infrastructure

Licence: partly MIT-licensed open source,
partly proprietary vendor software
Data access: open API to published content

Policy support *(optional)*

Funded by Wellcome Trust, HHMI,
Max Planck and Knut and Alice Wallenberg Foundation



Website
elifesciences.org



Reference Publications

About eLife: <https://elifesciences.org/about>

Organization Overview

eLife is a nonprofit organization that operates across three major areas: publishing, technology, and research culture (visit the website elifesciences.org). Its headquarters are in Cambridge, United Kingdom. The whole organization is run by 50 FTE employees as well as a team of practicing scientists on the journal; Editor-in-chief, Deputy Editor, Senior Editor, and Board of Reviewing Editor roles. As an open access life science journal publisher, eLife publishes important articles of the highest standards in all areas of biology and medicine and also explores new ways to improve assessment of research and publishing.

As of mid-Sept 2020, 819 articles with associated code have been identified and the code forked to the eLife GitHub repository.

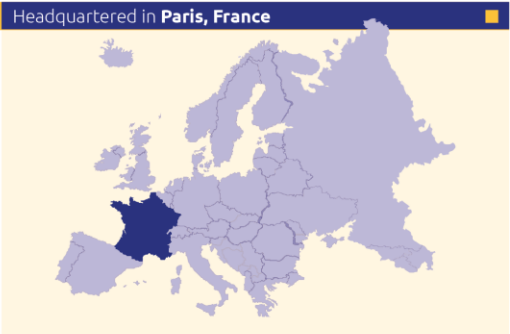
Scope

eLife is a disciplinary open access journal that addresses life sciences and biomedical researchers. The geographical scope is not limited; eLife operates internationally.

Workflow

Since 2017, eLife has been working on the concept of computationally reproducible papers. The open-source suite of tools that started life as the Reproducible Document Stack is now live on eLife as ERA, the Executable Research Article. Authors with a published eLife paper can register their interest to enrich their published work with the addition of live code blocks, programmatically-generated interactive figures, and dynamically generated in-line values, using familiar tools like R Markdown and Jupyter in combination with Stencila Hub's intuitive asset management and format conversion interface. The resulting new ERA publication will be presented as a complement to the original published paper. Readers of ERA publications will be able to inspect the code, modify it, and re-execute it directly in the browser, enabling them to better understand how a figure is generated. All changes are limited to an individual's browsing session and do not affect the published article, so anyone can experiment safely. Readers can also download the ERA publication – with all embedded code and data preserved – and use it as a basis for further study or derivative works. See example article, visit <https://elifesciences.org/articles/30274/executable>.





Category **Publishers**

Image Processing On Line (IPOL)

Open access research journal on reproducible algorithms

Type Not for profit	Year of creation 2009
Legal status Independent organisation	
Geographical scope International	
Content scope Disciplinary	
Software projects handled 500	
Source code archival Own storage (+ recently Software Heritage)	
Supported identifiers Extrinsic	
Curation Metadata: manual verification Source code: high level of review	
Estimated resources FTE: 18 Annual budget: ~100,000 € (yearly cost)	
Software infrastructure Licence: open source Data access: restricted API	
Website ipol.im	
Reference Publications The IPOL Demo System: a Scalable Architecture of Microservices for Reproducible Research. doi: 10.1007/978-3-319-56414-2_1	

Organization Overview

IPOL (Image Processing On Line) is an independent nonprofit open access journal (visit the website ipol.im). IPOL is headquartered in Paris, France, and was founded in 2009. Currently, the journal is run by 18 FTE employees and around 500 software items have been handled. The journal specifically addresses image and signal processing experts, such as researchers and mathematicians, but the service can also be used by non-experts. IPOL provides demos with all their published articles. To ease the work of the demo editor, IPOL created a Control Panel, a tool that easily creates and modifies new demos.

Scope

IPOL is a disciplinary open access journal which addresses researchers in the field of mathematics, algorithms, and especially reproducible research. The geographical scope is not limited, as IPOL addresses an international public.

Workflow

Since IPOL fully focuses on reproducible research, it only accepts free-software and open source code and releases all articles under a free documentation license. The languages supported by IPOL are C/C++, Python3, and Octave/MATLAB. The reviewing process by IPOL is more strict compared to many other journals. At least two reviewers are requested; one reviewer tackles the scientific part and the other checks the corresponding source code. The latter carefully checks if the pseudo code exactly matches the implementation, which is mandatory for publication through IPOL. Publications are managed through an Open Journal Systems (OJS) instance. The journal is indexed in SCOPUS, Emerging Sources Citation Index (ESCI) of the Web of Science, and the Directory of Open Access Journals (DOAJ). Recently IPOL started a collaboration with Software Heritage where the source code is currently indexed. IPOL operates at two levels of archival. First, published source code and article (PDF) are stored in their own servers under the same DOI. Because an object with a registered DOI cannot be modified, IPOL stores the history of all revisions. Second, data from users is archived in IPOL's own non-moderated archive of original data, which is useful for reproducibility and provides the possibility to recover and re-execute using different parameters. In regard to credit and attribution, IPOL acknowledges the authors of the article as well as the source code, the editor, and the editor that designed the demo. The quality of the published code is safeguarded by software guidelines.

Demo of Service: <https://vimeo.com/146499081> (French)



2.2.3 *Aggregators*

The following list provides the set of aggregators, including catalogues, that are represented in the SIRS Task Force.

- OpenAIRE
- scanR
- swMATH.org



These infrastructures are a respectable representation of the Aggregator subgroup landscape, as the services vary greatly in geographic scopes (national and international), scope in terms of content (universal and disciplinary), size, number of registered users, number of software projects handled, and workflow.



Category **Aggregators**

OpenAIRE

The European Open Science aggregator

Type	Year of creation
Not for profit	2019
Legal status	
Independent organisation	
Geographical scope	
International	
Content scope	
Global	
Software projects handled	
	200,000
Source code archival	
None	
Supported identifiers	
Intrinsic	
Curation	
Metadata: machine/human curation Source code: no	
Estimated resources	
	FTE:25
Software infrastructure	
License: open source (GPLA) Data access: open API http://api.openaire.eu	
Policy support (optional)	
National, Institutional and Global endorsement	
 Website	
Infrastructure website: www.openaire.eu Services website: explore.openaire.eu	
 Reference Publications	
https://www.liberquarterly.eu/articles/10.18352/lq.8110/ http://puma.isti.cnr.it/rmydownload.php?filename=cnr.isti/cnr.isti/2014-A0-034/2014-A0-034_1.pdf	

Organization Overview

OpenAIRE is an independent nonprofit company, headquartered in Greece. It was created in 2019, but has been operating as a Consortium since 2009. Currently, around 25 FTE employees are running OpenAIRE. It is an open science infrastructure for Europe that functions on two levels, networking for practices on open science and technical services. On the technical side OpenAIRE provides a number of services to bridge the thematic services and research infrastructures of the communities within the world of scholarly communication. It addresses the challenge of smoothening this process for the researcher and creates a common understanding of science, of course respecting the differences where they are. The first outcome of OpenAIRE is the guidelines for content providers. These are metadata profiles, based on standards, grown up with the communities themselves, with consultation. The profiles are continually updated as the concept of open science evolves. OpenAIRE produces guidelines for publication repositories, data, and software. The core service OpenAIRE provides is the OpenAIRE Research Graph. This graph contains metadata collected from trusted sources worldwide. Today, around 200,000 software items are handled and interlinked with scientific articles and other scholarly sources.

Scope

OpenAIRE functions on an international level, acting between different institutions and nations to ensure that best practices are shared between countries. Moreover, at a thematic level OpenAIRE works with different communities, mainly research infrastructures, trying to identify open science practices and roadmaps. However, OpenAIRE is also active at a global level, interacting with similar initiatives worldwide. OpenAIRE is a multidisciplinary metaportal for research output that is not limited to a specific domain in science.

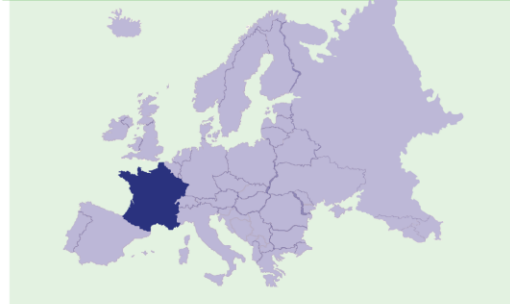
Workflow

For archival, OpenAIRE relies on Zenodo. Furthermore, OpenAIRE relies on existing PID systems and promotes them. Metadata about software is (i) collected (harvested) from scholarly sources and (ii) 'crawled' from software repositories when mining identifies scientific article full-text URLs to known software repositories. A PID to the persistent version of the software in Software Heritage is added. OpenAIRE created the Guidelines for Software Repositories to ensure credit and attribution. Data quality is delegated to data sources and managed via metadata harmonization and context-propagation.





Headquartered in **Paris, France**



Category **Aggregators**

scanR

French research aggregator

Type Public	Year of creation 2016
Legal status Hosted organisation	
Geographical scope National	
Content scope Academic	
Software projects handled 400	
Source code archival None	
Supported identifiers Intrinsic Extrinsic	
Curation Metadata: manual curation Source code: no	
Estimated resources FTE: 0.5 Annual budget € 100.000	
Software infrastructure Licence: open source data access: no API	
Policy support (optional) institutional endorsement	
Website scan.enseignementsup-recherche.gouv.fr	

Organization Overview

scanR was founded in 2016 and is maintained by the French Ministry of Higher Education Research and Innovation. The organization is headquartered in Paris, France, and is run by 0.5 FTE employees. As a search engine scanR contains four different types of objects; structures and entities, author details, fundings, and productions. Productions can be either a publication, PhD, patent, or software. As of July 2020, scanR has handled 400 software items.

Scope

The organization aims to be a research engine of national research with a focus on public funded research, but also private sector innovation.

Workflow

scanR filters all objects for output that are affiliated to France. The metadata of affiliations is very difficult to infer, especially for publications, because no open data sources with affiliations for publications exist. scanR aims to build an open data source containing this information in order to describe. Additionally, scanR builds links between the objects in such a way that all four objects that are represented in scanR are linked with each other. In order to achieve this, multiple IDs from different registries are used and linked. Articles are scanned for pieces of software by scanning for GitHub URLs. Using the URL, a link to Software Heritage is made. Also, the GitHub search engine is used to scan README files of pieces of software to check if a DOI is mentioned. If this is true, a link between the visited repository and the DOI that was mentioned is made. For archival, scanR uses existing archives, through Unpaywall data. Person identification is performed for credit and attribution, but is dependent on the sources that are harvested.





Headquartered in **Berlin, Germany**



Category **Aggregators**

swMATH

Mathematical software aggregator

Type Public	Year of creation 2011
Legal status Consortium	
Geographical scope International	
Content scope Disciplinary	

Software projects handled 33.341 (Sept. 2020)
Source code archival None

Supported identifiers Intrinsic: links to SoftwareHeritage, InternetArchive
Curation Metadata: human curation Source code: no

Estimated resources FTE:3 Annual budget € 100.000
Software infrastructure Licence: open source Data access: API in preparation

Policy support (optional)
Institutional and national endorsement

 **Website**
swMATH.org

 **Reference Publications**
swMATH:
<https://sinews.siam.org/Details-Page/swmath-a-publication-based-approach-to-mathematical-software>

Organization Overview

swMATH is a nonprofit information service for mathematical software that was founded in 2011 and headquartered in Berlin, Germany. The service is freely accessible and not only provides access to an extensive database of information on mathematical software, but also includes a systematic linking of software packages with relevant mathematical publications. The intention is to offer a list of all publications that refer to software recorded in swMATH. In particular, all articles included in Zentralblatt MATH (zbMATH) are listed. These can be articles that describe the background and technical details of a program, as well as those publications in which a piece of software is applied or used for research. In this way, swMATH provides information on actual use of the software that is otherwise impossible or very difficult to obtain. At the same time the documentation of literature referring to software is a valuable source of information for the authors of the software about where their software is used. If software is cited in scientific publications, this is also an important quality criterion, which is used by swMATH for software selection. As of September 2020, 33.341 software packages have been handled.

Scope

swMATH sees itself as a service to the mathematical community. The geographical scope is not restricted.

Workflow

New content is moderated ex-ante as only software from identified parties is accepted. The software is evaluated, while metadata is checked to ensure it conforms with the source code. The metadata is aggregated by editors and is curated through a semi-automated process. Credit and attribution relies on determination of the authorship via extraction from the reference article. The author order list from the publication is retained to indicate the author contribution, although in mathematics this is often in alphabetical order.

Demo of Service
<https://sinews.siam.org/Details-Page/swmath-a-publication-based-approach-to-mathematical-software>



3 STATE OF THE ART

3.1 Survey on Related Initiatives and Related Works

Software has played an essential role in research for decades and one can find a few long established initiatives in some research institutions like Inria (Alliez et al., 2020) or research communities like Computational Physics (Roberts, 1969), Numerical Computing (Press et al., 1986), Neurosciences (McDougal et al., 2016) and Astrophysics (Allen & Schmidt, 2015). The tidal wave of Free Source was also born in academia, before taking industry by storm as Open Source software. But it seems that general awareness about the importance of software *as a research output* has started growing only very recently, around 2010, in particular as a by-product of the reproducibility crisis (Barnes, 2010; Borgman et al., 2012; Colom et al., 2015; Konrad Hinsen, 2013; Rougier et al., 2017; Stodden et al., 2012) .

Without any pretension to exhaustivity, a few remarkable early signals of this awakening can be found for example in the Science Code Manifesto (Barnes, 2010), the creation of the INCF Software Center (Ritz et al., 2008), the creation of the Software Sustainability Institute in the UK in 2010, the creation of the IPOL journal in 2009 (Colom et al., 2015), and the first Artifact Evaluation introduced in 2011 for the ESEC/FSE Conference⁹ by Andreas Zeller with Carlo Ghezzi and Shriram Krishnamurthi, that is now widespread in Computer Science conferences (Childers et al., 2016; Krishnamurthi, 2011) and led to the approval of the ACM Badging schema (ACM, 2016).

Around 2015, a wealth of articles were already highlighting the importance of preserving and referencing software for reproducibility in many different areas and disciplines (Allen et al., 2017; Baker, 2016; Collberg & Proebsting, 2016; Gil et al., 2016; Krishnamurthi & Vitek, 2015), and initiatives were launched to start making a change, like the GitHub/Zenodo integration for archiving source code and registration of persistent identifiers and scholarly metadata (*Making Your Code Citable · GitHub Guides*, n.d.), the CodeMeta initiative (Jones et al., 2016), the creation of the FORCE11 working group on software citation (Smith et al., 2016), the DARTS artifact series (Wagner, 2017), and the RDA interest group on software source code (Gruenpeter et al., 2020).

While attention to *software* was only beginning to rise, the research community had moved forward at full speed on *research data*, to the point that the *FAIR principles for research data* (Wilkinson et al., 2016) were endorsed at the highest level during the September 2016 G20 meeting¹⁰. This chronology of events, and the fact that software was still largely seen as a tool, or just another piece of data, may explain why significant energy has been spent trying to see how software may fit into the FAIR principles, possibly with some minor changes (Gruenpeter et al., 2020; Katz & Clark, 2019; Lamprecht et al., 2020), instead of developing principles adapted to software anew.

Software development plays an essential role in research, so it is not surprising that for quite a long time in some countries there have been efforts to federate and support software developers working in the research community, like the DEVLOG network in France¹¹. More recently, the term Research Software Engineer (RSE) has been adopted by several national and multi-national initiatives in Europe and beyond that bring together individuals with skills in research software development, advocate for recognition of RSEs

9 See <https://www.microsoft.com/en-us/research/blog/new-award-honors-distinguished-artifact/>

10 See https://ec.europa.eu/commission/presscorner/detail/en/STATEMENT_16_2967

11 See <http://devlog.cnrs.fr/region>

and promote putting software on equal footing with other research outputs (de-RSE, NL-RSE, Nordic-RSE, UK-RSE, IS-RSE)¹².

In recent years we have seen software in general and software source code in particular finally start to be mentioned in policy documents, ranging from the French national plan for Open Science (Clément-Fontaine et al., 2019) to the Paris Call on software source code issued by the Inria-UNESCO expert group meeting (UNESCO Expert group meeting, 2019).

In this report, we focus specifically on three key components in the scholarly architecture for *software source code* used in research: *archives*, *publishers*, and *aggregators*. We provide below a short overview of relevant initiatives and works in these areas.

3.1.1 Archives

Pioneered by initiatives specifically dedicated to the history of computing, like the Computer History Museum and similar organizations, the activity of preserving *software programs* is a relatively recent concern in the history of archives, and it focused essentially on the archival of the *physical media* on which these executable programs were distributed (floppy disks, game cartridges, CDs, or DVDs), that were catalogued and stored exactly like books. A remarkable actor in this space is the French national library (BNF), as in France, unlike in other countries, software programs have been subject to legal deposit since 1992¹³. More recently, with the advent and commoditization of virtualisation and emulation technologies, the focus shifted to *keeping old software programs running on more recent machines*, in particular as a means to preserve access to digital assets¹⁴.

The interest in preserving *software source code* in its own right, though, is much more recent, despite having been identified as a crucial issue in a crystal clear seminal article by Len Shustek in 2006 (Shustek, 2006). Software source code has been largely considered outside of the scope of scholarly repositories and institutional and national archives until just a few years ago, when existing scholarly archives and repositories started to allow the deposit of source code bundles, that were assigned an extrinsic persistent identifier similar to those used for datasets: as an example, the first DOI (Paskin, 2010) registered for a software bundle at DataCite dates only from September 7th, 2011 (Fenner et al., 2018) and the *software category* was introduced in Zenodo only in 2014 and in HAL only in 2018 (Barborini et al., 2018; Di Cosmo, Gruenpeter, Marmol, et al., 2020). These scholarly repositories provide the deposited software bundles all the useful mechanisms already available for the other digital content they handle, like access control, metadata update mechanism, peer-review anonymous access, and optionally moderation or curation of deposit, as well as well-established interfaces with other repositories, like the OAI-PMH protocol.

One means by which software is preserved is via deposition in repositories such as Zenodo, and this process can be automated for software in GitHub that is formally released. According to statistics from (Fenner et al., 2018), the largest source of DataCite DOIs for software is from software in Zenodo. However, software deposits in the form of source code bundles have been made available in various forms in different scholarly repositories or digital archives that want to have control of their own data, like the many distribution platforms mentioned below. Meanwhile, software development has been growing exponentially over the past half a century, and the tools and platforms that support it have been evolving at a fast pace, with *original content doubling every 22 months*, and *original*

¹² See <https://researchsoftware.org/> and <https://society-rse.org/about/history/>

¹³ See <https://www.bnf.fr/le-depot-legal-numerique>

¹⁴ See the PERSIST project <https://unescopersist.org/> and the various projects of the [Software Preservation Network](#) for more on this subject.

commits doubling every 30 months, as shown in Figure 2, taken from the broadest analysis of software development evolution of which we are aware (Rousseau et al., 2020).

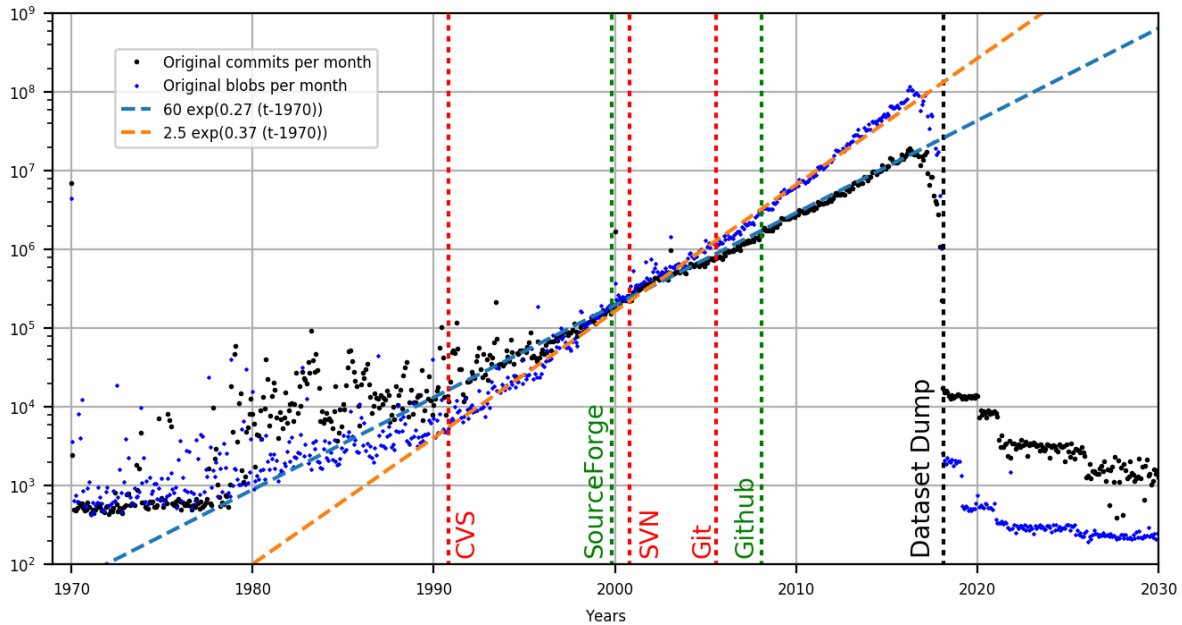


Figure 2. Global production of original software artifacts over time, in terms of never-seen-before revisions and file contents (lin-log scale). Major events in the history of version control systems and development forges are materialised by vertical bars.

For decades, software source code has been made available by software developers worldwide essentially through platforms that fall mainly into one of two categories:

Collaborative development platforms

Services that allow the creation and collaborative development of software projects; a few well-known examples are SourceForge, Gitlab.com, GitHub, and BitBucket.

Distribution platforms

Services mainly used to distribute (versions of) software packages; a few well-known examples are CPAN, CRAN, CTAN, PyPI, Maven Central, and NPM.

The sudden shutdown of huge development platforms like Google Code and Gitorious.org in 2015¹⁵, endangering over *one million and a half software projects*, brutally reminded all of us that, surprising as it may seem, none of these platforms was primarily designed as an *archive*, meant to preserve for the long term the software source code together with its development history. These events exposed the urgent need to develop a *universal software source code archive* with the mission to collect, preserve, and share the source code of *all software publicly available, with all its development history*. As a result of a combination of circumstances, infrastructures with precisely this mission had started to be built a few months before, by Software Heritage, an open, non-profit initiative launched by

15 See <https://www.softwareheritage.org/2016/09/01/google-code-content-now-safely-collected-in-software-heritage/> and <https://www.softwareheritage.org/2016/07/21/gitorious-retrieved/>

Inria in collaboration with UNESCO (Abramatic et al., 2018). All those projects were salvaged in time.

Software Heritage is the only infrastructure designed specifically to preserve in the long term *all software source code with its entire development history*, using the same technology incorporated in modern distributed VCSs like Git or Mercurial (Di Cosmo et al., 2018; Di Cosmo, Gruenpeter, & Zacchiroli, 2020; Di Cosmo & Zacchiroli, 2017), and it provides simple and generic mechanisms for archiving and referencing source code in scholarly publications (Di Cosmo, 2020b).

3.1.2 Publishers

Since Buckheit, Donoho, and others warned about the *credibility crisis* in scientific research (Buckheit & Donoho, 1995), the problem of reproducibility has been confirmed by thousands of scientists from diverse fields (Baker, 2016). This problem is common to many research disciplines, but it is especially relevant in computational sciences, where in many cases the result of the research is an algorithm and therefore there is no excuse not to address reproducibility. From the starting case study of Wavelab (Buckheit & Donoho, 1995) several other works have addressed this topic from the perspective of repeatability in computer systems (Collberg & Proebsting, 2016), and its relation with research contributions (Benureau & Rougier, 2018). Note that this problem is not particular to computational sciences, but to all kinds of research which involves any kind of side computing. For example, publications on social studies or journals on humanities which might write their own software to collect data and generate figures to present results.

Several initiatives have been developed to try and provide solutions to this crisis, at different levels. Peer review of the software artifacts has been introduced, via artifact evaluation committees and badging schemas (ACM, 2016; Childers et al., 2016; Krishnamurthi, 2011), and a variety of solutions to archive and reference these artifacts have been offered by publishers, either via their deposit as ancillary material associated with the research article (ACM, n.d.), or via publication of software or evaluated research artifacts in their own right (Seinstra et al., 2015; Smith et al., 2018; Wagner, 2017). Pioneering journals such as IPOL journal (Image Processing On Line) (Arévalo et al., 2017; Colom et al., 2015) went much further, building infrastructures to run, evaluate, and compare algorithms, in the specific field of image processing.

The holy grail of *full reproducibility of the executables* associated with research articles has been the object of quite a lot of attention in many disciplines, with a large variety of tools developed over time, ranging from virtualization or containerization environments, to notebooks, to fully executable research articles, see e.g. (Konrad Hinsén, 2020; Spagnuolo & Veltkamp, 2013). A critical survey of the existing solutions and their advantages and limitations is out of the scope of this report, but we stress here that these efforts are *complementary* to the need to properly archive and reference the full corresponding source code, and do not supersede it.

Several initiatives move towards *augmented publications* (Barnes, 2010; Clément-Fontaine et al., 2019; Colom et al., 2019), when the source code and data is a fundamental piece of the publication, being part of the research work itself by combining text, source code, and data as a whole (Borgman et al., 2012), and giving to the authors of software source code the credit they deserve (Alliez et al., 2020; CASRAI, 2015; Di Cosmo, Gruenpeter, & Zacchiroli, 2020; Smith et al., 2016). This implies changes in the way research is evaluated (Allen et al., 2017; Childers et al., 2016), given that these items are considered a significant part of the research work (UNESCO Expert group meeting, 2019). These new kinds of publishers are encouraged to work tightly with open repositories (Bönisch et al., 2012) and permanent archives (Abramatic et al., 2018; Di Cosmo, Gruenpeter, Marmol, et al., 2020) as well as exchanging artifact metadata automatically with standardized schemas (Burton et al., 2017).

3.1.3 Aggregators

Aggregators for research software collect metadata information about software – and in some cases also the source code – such as persistent identifiers, download URLs, abstracts, contributors, links to other research products, etc. Information is collected from a variety of different sources ranging from reference articles to manuals, repositories, and even other aggregators.

Research software-focused aggregators Aggregators for research software in specific disciplines such as the swMath service for Mathematics with more than 30.000 curated entries (Bönisch et al., 2012) or the ASCL service for Astrophysics with over 2.000 curated entries (Allen & Schmidt, 2015) have been around the longest, but new and lively initiatives came to life more recently, like the Papers with code initiative in Machine Learning with over 34.000 entries (Stojnic et al., 2019).

In order to ensure the quality of the information collected, other initiatives rely on a sort of editorial board, like ASCL for Astrophysics (Allen & Schmidt, 2015). Another remarkable example is the catalog built by the Plume project in order to collect information about software that is useful for research activities (The Plume Team, 2013): it maintains a collection of over 400 entries manually curated about software projects that are successfully deployed and in use in at least three different research laboratories.

Research software in-context aggregators Software specific aggregators like the one mentioned above focus particularly on the curation of a collection of software entries, but software has been attracting attention also of aggregators that were not specifically designed or built to address software in the first place.

A remarkable actor in this category is DataCite, a DOI registration agency for scholarly content that has included software in its metadata schema since 2011¹⁶ and that works with scholarly repositories like Zenodo that provide DOIs for their deposits. After the publication of the FORCE11 software citation principles (Smith et al., 2016), it further updated the schema in 2017, specifically focusing on research software (DataCite Metadata Working Group, 2017), and as of October 2020, it counts approximately 150.000 DOIs registered for different software versions of approximately 40.000 different software projects, mostly via the Zenodo/GitHub integration (Fenner et al., 2018). The main focus of this effort is on aggregating in a single registry, the DataCite Metadata Store, all the references to software products, to include them in the DataCite Commons (Fenner, 2020), or to make them available to other generalistic aggregators, like OpenAIRE, in the DataCite metadata format, or in other formats, including CodeMeta or schema.org (Burton et al., 2017), leaving curation and quality control to other actors.

While most aggregators of scholarly publications and metadata, like CORE (Knoth, n.d.) or Google Scholar, focus on scholarly publications, others are increasingly including in their underlying information models other research entities such as datasets and software. For example the OpenCitations initiative has expressed interest in broadening its scope to cover software citations (Peroni & Shotton, 2020) while OpenAIRE is already harvesting software related metadata to link it to other scholarly resources and actors into the OpenAIRE Research Graph (Manghi & Bardi, 2019). Today, OpenAIRE counts 198.000 software entities harvested from Biotools, Zenodo, Figshare, DOE CODE, EGI Application Database, DataCite sources, which are in turn linked to SoftwareHeritage persistent

¹⁶ See for example the publication (Liang & Kai Yong, 2013) referencing a software package used to generate data for the publication (Liang & KaiYong, 2013).

identifiers and original software repositories (e.g. GitHub, BitBucket, etc.). Interestingly, OpenAIRE has identified (via harvesting and full-text mining) semantic links between publications and software and ~1.850 links between research projects and software.

Added-value of aggregators The added value of aggregators is twofold. On the one hand the ability to integrate, harmonize, and offer access to information originating from different sources, which should otherwise be accessed independently. On the other hand, the ability of enriching the aggregated content with information that was not available at the original sources. For example, a variety of approaches have been used to link software with the relevant research literature: while some workflows fully rely on manual submission and curation, others also use tools of various degrees of sophistication to help extract references to software from the scholarly literature in the discipline. Due to the diverse forms that a mention of software can take in a scholarly work, fully automated extraction of references is far from being sound and complete (Howison & Bullard, 2015), and there are different experiments ongoing. For example, the Asclepias project relies on domain-specific and generic aggregators like PubMed, NASA ADS and CrossRef/DataCite to extract references to software (Muench et al., 2017), while Papers with code proposes to use natural language processing and machine learning to help a community effort to build this correspondence (Stojnic et al., 2019), while OpenAIRE exploits full-text mining to identify URLs to known software repositories (e.g. BitBucket, GitHub, etc.) into article PDFs and adds them as relationships into the article metadata, together with the related persistent URL to SoftwareHeritage.org.

These approaches use research publications as a source of truth to identify relevant research software, and can produce quality valuable information when leveraging the quality curation process of the research publications, as is in particular the case of swMath (Bönisch et al., 2012). Their workflow could be simplified if the publisher directly included, in the metadata of the article, a persistent identifier for the software projects used, produced or mentioned in the publication, with appropriate description of the level of granularity at which the software project is mentioned. This information could be included in the metadata sent by the publisher to Crossref, or the JATS metadata deposited by the publisher in full-text archives such as PubMed Central¹⁷.

Last, but not least, the growing interest in software from aggregators brings with it at the same time great hopes for improving the practice of Open Science, and great challenges due to the potentially disruptive effects that the inevitable emergence of metrics on software production or citation in the scholarly world might have. On the one hand, the quality of some of the metrics that have been proposed, e.g. in the Open Science Monitor, has raised major concerns (Dacos et al., 2018). On the other hand, one would need to carefully consider all the implications before promoting purely numeric indicators for software, especially now that a growing international consensus is emerging around principles that value qualitative criteria over quantitative ones, like DORA (*San Francisco Declaration on Research Assessment (DORA)*, 2013) or the Hong Kong principles (Moher et al., 2020).

3.2 Summary of the State of the Art Presentations in the Group

In this section we summarise the key findings that emerged from the exchange sparked by the presentations of the practices of each participant infrastructure.

¹⁷ The Force11 Software Citation Implementation Working Group has undertaken work in this area.

3.2.1 Archives

Archiving research software is of importance to ensure that research artifacts are not lost. At present, various archives with specific discoveries, such as metadata and PID standards, and different archiving strategies are available for open science and are partly interlinked.

Software Heritage assembles not only research software, but source code in general. It makes use of systematic harvesting, so no explicit deposit is needed, and uniformly represents all VCSs, with provenance and traceability. Intrinsic identifiers for software are used systematically for the over 20 billion software artifacts in the archive, covering all levels of granularity, such as project status, project release, state of source code, and code fragment.

HAL, on the other hand, is specific to research software and requires deposit. The service has extensive, software-specific, metadata and involves human intervention for careful and manual curation of metadata. Extrinsic identifiers are assigned to the metadata, but intrinsic identifiers are assigned, via transfer to Software Heritage, to the software source code itself.

Zenodo assembles mainly research software. Like HAL, a deposit is required, however deposit automation is possible for users of GitHub that explicitly enable synchronisation with Zenodo. Metadata can be edited by the owner, and support for more advanced curation processes is planned. Extrinsic identifiers (DOIs) are assigned to both the project as a concept, and the software specific release. Integration with Software Heritage is planned.

Given the diversity in existing software archives, we consider that there is no urgent need for new infrastructures, but interconnecting the above-mentioned archives and repositories and the many others that exist or will come into existence should be prioritized. Next step is to expand the functionalities of existing infrastructures to, for example, automate metadata extraction, harmonize software metadata standards, support human curation of metadata, and support metrics. Guidelines for researchers should explicitly mention deposit and archival as an important issue.

3.2.2 Publishers

Over the past few years several publishers have led the effort in the transition towards open access as the predominant model of publication for scholarly outputs. This also paves a path for fair and affordable conditions from the start for the dissemination of software, but support for software outside of specialist journals is still limited. The participating infrastructures reflect a large variety of scopes and strategies, as seen in today's publishing business.

Dagstuhl Publishing, instead of explicit support for software in general, focusses on artifacts supplementing the textual contribution. The software source code is archived using Dagstuhl's own storage. Software artifacts are published separately from the related paper, with their own metadata and DOI. All software artifacts pass an artifact evaluation and metadata is manually curated.

The eLife journal has implemented open science and reproducibility standards that focus on use and re-use of software as well as giving credit to software authors. The source code generated for an article is expected to be licensed under a permissive license and eLife archives it to the eLife GitHub repository. eLife applies the FAIR principles for software citations. However, citing software is not common in the Life Sciences sector yet so schematron (a rule-based validation language for making assertions about the presence or absence of patterns in XML) is employed during the production process to search the XML content for software mentions - if the author has not referenced correctly, they are asked

to do so. The software metadata is not curated, but quality checks are performed to ensure the elements required to build a citation are present.

IPOL accepts only free software and open source code and releases its articles under a free documentation license. The source code is reviewed in detail, focusing on the mathematical details of the algorithms and checking that the implementation matches accurately the pseudo-code descriptions in the article. The source code is archived in Software Heritage, and stored in its own infrastructure. The article and corresponding source code are stored under the same DOI, and software source code is provided with the SWHID intrinsic identifier. The metadata is curated via automated verification and credit is given to all authors and editors of the article, software, and demo editors.

Given the great diversity in strategy, scope and resources of different publishers, there is a need for a low-barrier-to-entry standard that is easily implemented by all, while allowing for higher levels of curation to be implemented by some of them (see point (6) in Section 4.1.4 Cite/Credit for details). Also, as authors do not yet understand what is expected from them to support the four pillars (see Section 2.1.1 Archive, Reference, Describe, Credit: The Four Pillars), there is an opportunity for publishers to educate authors on the necessity of sharing software source code and encourage a standard workflow.

3.2.3 Aggregators

Aggregators collect, curate, select, present, and aggregate information about research software from various sources to improve findability in diverse communities. In general, the information space data model describes a scientific knowledge graph, whose nodes and edges conform to known research-related entity types (e.g. publications, data, software, authors, and organizations). The kind of information collected, the target data sources, its post-processing, and the data model of the resulting information space, depends on the target use-case application. Typical applications in the scholarly communication domain are *discoverability* (e.g. catalogues), *usage statistics*, *reproducibility*, *research impact* (e.g. citation indexes), etc. Two main aspects characterise scientific knowledge graph aggregators (https://doi.org/10.1007/978-3-030-55814-7_16):

Collection of information Aggregators generally collect metadata records, describing digital or real-world objects, and in some cases the digital object themselves, e.g. scientific articles (OpenAIRE¹⁸, zbMATH¹⁹, BASE, Google Scholar, Semantic Scholar²⁰), research data (e.g. Elsevier Data Search, Google Dataset Search).

Post-processing of collected information Aggregators post-process the information collected to build the intended information space. Several functional challenges arise, related with mapping heterogeneous exchange formats, structure, and semantics onto a common internal representation of the information space data model, but also with deduplication and creation of identifiers (intrinsic or extrinsic). Also, non-functional challenges are an issue, related to storage and processing capacity/sustainability. Aggregators are then characterised by their specific *data curation process* ensuring quality of and added-value to the collected data, i.e. by mining, crawling, inferring, editing, validating, etc.

As examples of aggregators targeting research software we report:

swMATH not only provides access to an extensive database of information on mathematical software, but also includes systematic linking of software packages with relevant

¹⁸ <https://www.openaire.eu>

¹⁹ <https://www.zbmath.org>

²⁰ <https://www.semanticscholar.org>

mathematical publications (Bönisch et al., 2012). New content is moderated ex-ante as only software from identified parties is accepted. The software is evaluated, while metadata is checked to conform with the source code. The metadata is aggregated by editors and is curated through a semi-automated process. Credit and attribution rely on determination of the authorship via extraction from the reference article. Whenever possible, a link to the code archived in Software Heritage is provided.

The OpenAIRE Research Graph is an open and transparent metadata collection bringing in 12,000 trusted scholarly communication sources worldwide, whose content ranges from publications, datasets, and software to funders, projects, organizations, authors, and information sources. The Graph data model relies on existing PID systems and promotes these, but also hosts URLs, cool URLs²¹, and other local identifiers (from institutional/thematic repositories, e.g. ArXiv, EuropePMC, etc.). With respect to research software, descriptive metadata is collected from scholarly sources, e.g. institutional repositories, research software repositories (e.g. EGI AppDB, DEO-CODE), and crawled from software repositories when, as a result of data mining article PDFs, a link from an article to a software repository is identified. Data quality is delegated to data sources and managed via metadata harmonization and context-propagation. As an orthogonal but key activity for software metadata aggregation, OpenAIRE created the Guidelines of Software Repositories²², a metadata profile based in DataCite that focuses on Research Software.

ScanR is a search engine that is specific to research productions, which can be in the form of a publication, PhD, patent, or software. ScanR interlinks objects that are associated with each other using IDs from different registries. Articles are scanned for pieces of software by scanning for GitHub URLs and, if found, linking to Software Heritage. Currently, the metadata is not curated.

3.3 *Best Practices and Open Problems*

Following the description of the current state, best practices were identified that the TF believes should be implemented in the ideal Scholarly Infrastructure for Research Software, supporting the future EOSC.

The section covers the best practices for all four pillars (see Section 2.1.1 Archive, Reference, Describe, Credit: The Four Pillars). For each best practice, we indicate which pillar is concerned, what the current status of the best practice is, what gaps should be addressed, and what kind of action needs to be taken to overcome the gaps (research, development, deployment, adoption). Note, any concerns that apply to the three representative groups are addressed in Section 3.4 Cross-cutting Concerns and not repeated in the tables of each group.

3.3.1 *Best Practice Principles for Archives*

The workgroup on archives identified several gaps that need to be addressed to aid the ideal infrastructure. First, as one does not need to reinvent the wheel, the archival community should agree on an overall architecture to integrate existing infrastructures. Next, the software archiving workflow needs automation and should be integrated with the development platforms. Automation of the workflow will be facilitated by standardisation and harmonisation of different processes within the workflow, mentioned in the table below. Additionally, software citations should be promoted and metrics should be supported. Last, the ideal architecture interconnecting a variety of infrastructures for research software needs inclusiveness of archives for both open software, as well as non-open software, and the ability to ensure the universal archival and reference of the source

²¹ See <https://www.w3.org/TR/cooluris/>

²² <https://software-guidelines.readthedocs.io/en/latest/>

code of all software, not just research software. This clearly leads to an architecture with a Universal Software Archive that archives all publicly available software source code, and a variety of scholarly repositories that connect with the Universal Software Archive on one side, and offer services specific to the academic community on the other side.

Addressing the above-mentioned gaps, the following best practices were identified:

Best practice	Pillar	Status	Gaps	Priorities
Inclusiveness of archive (also non-open and non-research software)	Archive	Software Heritage (building the Universal Software Archive), and a broad spectrum of scholarly repositories	Interconnection	Development
Automation of software archiving workflow and integration with development platforms	Archive	Preliminary work done	Cover the long tail of development platforms	Development
Software-specific category supported in search filters	Archive Description	Different implementations available	Harmonization	Adoption
Identifiers for software projects and artifacts	Reference	Multiple solutions available (see (Gruenpeter et al., 2020))	Clear guidelines on when to use intrinsic and/or extrinsic PIDs	Research Harmonization Adoption
Explicit tracking of versions of software artifacts	Reference Description	Different approaches available (depending on philosophy)	Better understanding of how to track different kinds of versions, in Software Heritage and in scholarly repositories	Harmonization
Explicit tracking of versions of metadata	Description	Some archives keep this information internally	Make this information available externally	Development
Metadata moderation and/or curation	Description Credit	Some repositories implement specific workflows	Harmonization of practices, and implementation of the workflows	Research
Metadata licensing with aim for an open license CC0	Description	Implemented	Awareness	Adoption
Guidelines for software archiving	Description Credit	Multiple available: depends on the goal sought	Harmonization	Research
Support software citation	Credit	Recommendations are available at different levels	Quality of metadata (attribution in particular)	Research Development

			Proper bibliographic styles (bibtex entries only recently made available for BibLaTeX)	
--	--	--	--	--

3.3.2 Best Practice Principles for Publishers

A lot of variety is found between what each publisher does, and little support for software is found outside specialist journals. Also, authors do not yet understand what is expected of them to support the four pillars (see Section 2.1.1 Archive, Reference, Describe, Credit: The Four Pillars). These are some of the open problems identified by the subgroup working on journals.

To overcome these issues, the following best practices were identified:

Best practice	Pillar	Status	Gaps	Priorities
Support and guidance for authors regarding referencing and archiving source code during submission	Archive Reference Describe Credit	Sometimes addressed, but costly and time consuming	Education and training	Development
Use of VCSs to write code	Reference Describe Credit	Code hosting platforms are not archives	Awareness of code hosting platforms	Adoption
Automatic posting of code to an archive on publication	Archive Reference Describe Credit	Not yet adopted	Awareness of differences and links between code hosting platforms and archives	Adoption
Simple to use tools or guidelines to cite and reference software produced or used	Credit	Requirements understood, but development in infancy	Tool availability Education	Development
Quality control / peer review of artifact in conjunction with associated article/dataset	Credit (reviewer) Describe	Status limited to Computational Sciences discipline	Recognition of the value of this (resource to support this limited) Lack of multidisciplinary skills (software engineering and scientific discipline)	Adoption
Submission procedure regarding code and software	Credit Describe Reference	No standard yet	Lack of governance Education	Development Adoption

3.3.3 Best Practice Principles for Aggregators

Aggregators of research software information are hindered in their efforts to provide a proper open software infrastructure by foundational issues concerning the identification, description, and exposure of research software. The tools typically used to create and maintain software are not always designed to support the needs of scholarly communication, and a lack of agreement on common practices has led to the use of a variety of metadata schemes and a misalignment between the metadata associated with code repositories and that required for scholarly purposes. Inconsistent use of PIDs complicates the unique identification of software and the proper attribution of credit.

Building aggregators for software is consequently a painful process — even what constitutes research software is not straightforward to define. The following best practices have been identified to address these issues:

Best practice	Pillar	Status	Gaps	Priorities
Software defined as a specific class at similar level of datasets and article publications	Archive Reference Describe Credit	Software is not always described as separate entity	Not always possible to describe granularity correctly	Deployment
More disciplinary discovery resources	Archive Reference Describe Credit	Some exist (e.g. swMATH, BioTools)	Similar approaches should be advocated in other communities Offer solutions as a “package” (workflows, policies, tools) to facilitate diffusion	Definition
Ensure each software entry has a link to a long-term archive	Archive Reference	Embryonic	Lacking adoption	Adoption
Promote usage of PIDs	Reference	Several established systems of identifiers are available (Gruenpeter et al., 2020)	Lacking adoption	Adoption
Interlinking software (via PIDs) to other research entities (e.g. publications, datasets, services, authors)	Describe Credit	Some metadata formats provide possible solutions (e.g. DataCite, CodeMeta, Scholix)	Agreed-on practice/semantics is missing	Definition, endorsement, and adoption

Promote CodeMeta	Describe Credit	Source code may be found in several places, with different possibly inconsistent or conflicting metadata	Common metadata standard Crosswalks between different software metadata standards Missing engagement with repository platforms to establish metadata frameworks for software	Adoption
Metadata collection protocols (APIs)	Reference Describe	There is no general agreement on which protocols should be used to share research software metadata and code	Aggregators need to face the complexity of supporting (and maintaining) code to collect metadata (and code) information about software via different protocols (and formats)	Definition, endorsement, and adoption

3.4 Cross-cutting Concerns

This section presents the cross-cutting concerns identified looking at the results of the subgroups working on archives, journals, and aggregators/catalogues. For each item, we try to indicate possible ways of addressing it, either through existing solutions, or via future work.

3.4.1 Metadata

Proper description of software artifacts is needed across the line. It is essential for the four pillars (see **Error! Reference source not found.**) in FAIR. For metadata, the following common requirements have been identified:

What	Candidate solutions	ARDC	FAIR
Machine readable, standard format	schema.org ²³ , CodeMeta ²⁴ , SPDX ²⁵	D	F
Roles for authors/contributors	(Alliez et al., 2020)	C	

²³ <https://schema.org/>

²⁴ <https://codemeta.github.io/crosswalk/>

²⁵ <https://spdx.dev/specifications/>

Licence information (metadata)	CC0	D	R
Licence information (software)	SPDX license list ²⁶	D	R
Linking to other research outputs	ensure appropriate terms are in the metadata schema (e.g. referencePublication for publications in CodeMeta) with appropriate identifiers for the other research outputs (e.g. DOIs for publications)	D	FIR

3.4.2 Identifiers

Proper identification of software artifacts is needed across the line. It is essential for the R(eference) and C(ite) from the four pillars (see Section 2.1.1 Archive, Reference, Describe, Credit: The Four Pillars), and for the A(ccessible) in FAIR. The following common requirements have been identified:

What	Candidate solutions	ARDC	FAIR
intrinsic/decentralised for reproducibility	SWHID ²⁷	RC	FA
support versions and all sw granularities	various (intrinsic and extrinsic)	RC	FA
persistent (for extrinsic identifiers)	various	RC	FA
standardised	various (intrinsic and extrinsic)	RC	FA

For an introduction to intrinsic and extrinsic identifiers, see this dedicated blog post²⁸. For a deeper analysis, see (Di Cosmo et al., 2018; Di Cosmo, Gruenpeter, & Zacchiroli, 2020).

3.4.3 Quality and Curation

At various degrees, the issue of the *quality* of the *metadata* about software and/or of the *software itself* emerges for all actors. The following facets of quality have been identified:

What	Candidate solutions	ARDC	FAIR
Deduplication of software source code	Intrinsic identifiers (SWHID ²⁹)	RC	
Human curation of metadata	Various (see HAL ³⁰ , swMath ³¹ , ...)	DC	FR
Evaluation of software source code	Various (see IPOL ³² , AEC ³³ , DARTS ³⁴ , ...)	DC	
Plagiarism detection	Manual inspection, SWH scanner	C	

²⁶ <https://spdx.org/licenses/>

²⁷ <https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html>

²⁸ <https://www.softwareheritage.org/2020/07/09/intrinsic-vs-extrinsic-identifiers/>

²⁹ <https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html>

³⁰ <https://dx.doi.org/10.2218/ijdc.v15i1.698>

³¹ https://link.springer.com/chapter/10.1007/978-3-662-44199-2_103

³² <https://www.ipol.im/>

³³ <https://www.artifact-eval.org/>

³⁴ <https://www.dagstuhl.de/en/publications/darts/>

	(forthcoming)		
Provenance, source of authority	Keep information about who produced/curated the metadata, how, and when; this may include a reference to articles that describe the algorithm or the artifact itself; keep the history of modifications of the metadata	DC	R

Since not all stakeholders/disciplines are able/willing to implement all these levels of curation and quality control, one should provide a clear indicator of the quality control/curation level implemented. A possible idea to look at is the ACM Badging³⁵ schema.

3.4.4 Metrics

As for all research outcomes, it will be useful to provide metrics to cater to the needs of a variety of actors for software too. The group agrees that these metrics should be *open, verifiable, and shareable*.

Nonetheless, scholarly indicators about software *cannot be simply reduced to the number of citations* of a particular software (or version of) in the scholarly literature, for a variety of reasons. First, there is still *no widely adopted standard* to cite software: surprising as it may seem, while TeX and Bib(La)TeX have been standard tools for more than a generation of researchers, a full-fledged bibliographic style supporting software as a first class citizen was made available for BibLaTeX users only in 2020 (Di Cosmo, 2020b). Second, important software libraries on which many research software depend *may not be cited directly* (Chawla, 2016; Zhao & Wei, 2017). Third, research software may have a significant *impact outside of academia*, where scholarly citations simply do not count. Last, but not least, *the value* of the contribution contained in a piece of a software *does not necessarily relate to its popularity, or amount of reuse*³⁶.

For all these reasons, we believe that a variety of metrics need to be developed, and properly assessed for their quality and impact, before being promoted widely. To this end, it will be necessary to bring together a broad spectrum of expertise, and include in the conversation representatives of the research community that will be directly impacted by the creation of these metrics.

3.4.5 Guidelines

A general need for actionable, standardised guidelines is seen across the line:

- for researchers/developers that self-archive software
- for researchers/developers that submit software in a publication workflow
- for reviewers/moderators that curate software metadata
- for reviewers/moderators that evaluate software itself
- for publishers that handle software in their publication workflow

³⁵ <https://www.acm.org/publications/policies/artifact-review-badging>

³⁶ Here is a very well-known example of an extremely popular tiny piece of source code <https://arstechnica.com/information-technology/2016/03/rage-quit-coder-unpublished-17-lines-of-javascript-and-broke-the-internet/>

Over the past years, several individuals, institutions and working groups have been documenting existing processes or proposing new ones (Alliez et al., 2020; Di Cosmo, 2020b; Di Cosmo, Gruenpeter, Marmol, et al., 2020; Gruenpeter & Sadowska, 2018; Katz et al., 2020; Smith et al., 2016). These approaches, of various degrees of generality and maturity, should be compared and tested, integrating feedback from the research community.

3.4.6 Tools and Workflows

Guidelines have usually little effect without proper tools to support them. Here are a few shared concerns about these tools:

- *added value for the researcher*: any action demanded on the researcher side should provide immediate value for the researcher themselves.
- *automation*: of tasks that do not require human intervention (e.g. triggering archival of new software releases).
- *avoid duplication*: the same information *should not be entered* in different formats/places.
- *information preservation*: tools and workflows should manipulate *machine readable* information and *preserve it along the way*.
- *validation*: metadata should be *validated early with regard to the specified schemas*. As an example, this is the approach taken in the CodeMeta generator tool³⁷ contributed by Software Heritage.
- *separation of concerns*: classical notions of separation between data model and presentation layer apply. See for example the preparatory work³⁸ done to produce the biblatex-software package³⁹ that supports software citation.

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

³⁸ <https://gitlab.inria.fr/qt-sw-citation/bibtex-sw-entry/-/blob/master/README.md>

³⁹ <https://ctan.org/pkg/biblatex-software?lang=en>

4 THE ROAD AHEAD

In this section we present the general requirements for the architecture of interconnected scholarly infrastructures supporting archival, reference, description, and citation of (research) software source code. The special uppercase terms like MUST and SHOULD have the specific meaning defined in RFC 2119 (Bradner, 1997), and recalled in Section 6.1 Glossary.

The high-level view of the architecture is depicted in Figure 3.

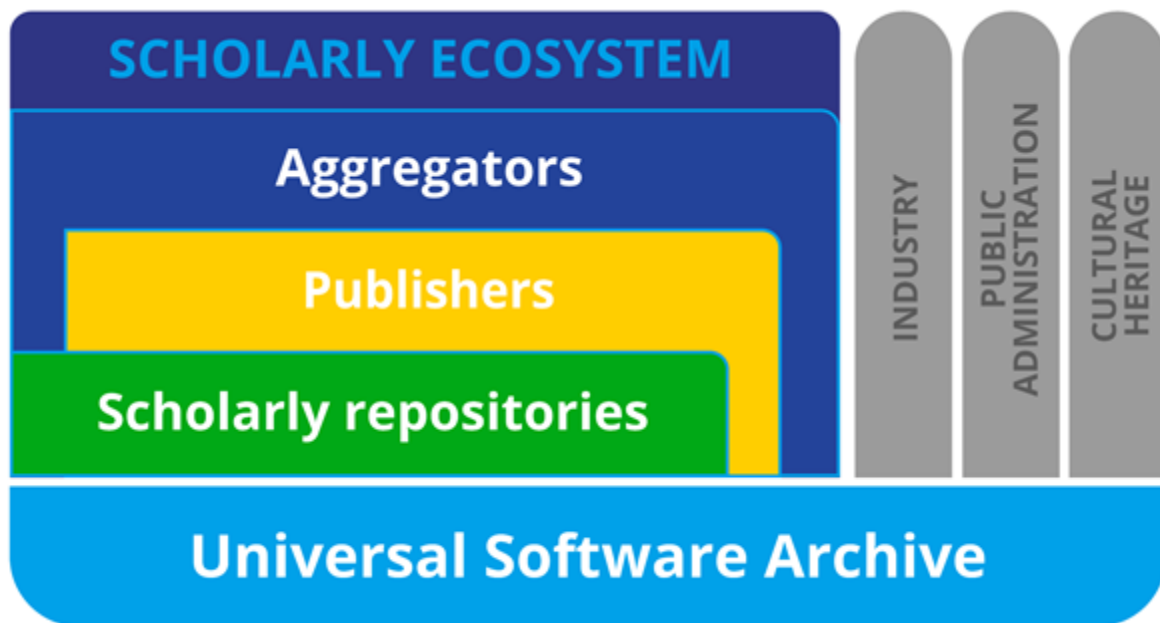


Figure 3. Architecture of interconnected scholarly infrastructures supporting archival, reference, description and citation of (research) software source code.

The base layer of the architecture is the Universal Software Archive, that targets all software source code and not just the source code deposited through the scholarly repositories. This is necessary to deal with the specificities of the software world, where software source code developed by researchers is only a thin layer on top of a complex web of tightly intermixed components and dependencies (see Section 2 Introduction and Figure 1). It also allows to share standards and approaches, as well as the necessary efforts, with the broader software development and preservation ecosystems, that includes a variety of actors, ranging from industry to public administration and cultural heritage.

This Universal Software Archive is built and maintained by Software Heritage, an international non-profit open organization started by Inria in 2015, in collaboration with UNESCO: it brings together a growing, broad spectrum of stakeholders, ranging from industry to academia and public administrations, that support its mission to collect, preserve and make accessible for the long term, with its complete development history, the source code of *all software publicly available*; see (Abramatic et al., 2018; Di Cosmo & Zacchiroli, 2017) for a presentation of the approach and principles behind Software Heritage.

A variety of different ecosystems are building applications on top of this basic layer, ranging from industry (Yates, 2019) to public administration (DINUM, n.d.) and cultural heritage (Bussi et al., 2019). In this report, we focus on the scholarly ecosystem, with scholarly

repositories where research software may be deposited explicitly, publishers that may link publications with the source code of the associated software, and aggregators that offer researchers a broader view of the available information.

4.1 General Requirements

Components needed to implement basic ARDC functionalities, and their interactions.

4.1.1 Archive

Objective

Support reproducibility, verifiability, and reusability of research results:

- ensure *research software artifacts* are preserved in the long term
- ensure *source code of all their dependencies and associated tools* are also archived (K. Hinsén, 2019)

Components

1. Universal Software Archive: specifically designed for software source code
 - *proactive* archival of *all* software source code (including all dependencies of research software)
 - *faithful representation* of the complete history of development in the original VCSs, including:
 - commits
 - releases
 - tags
 - fork and merge operations
 - their associated metadata (commit messages, etc.)
 - ability to *trace software provenance* across multiple projects
 - *export* of any software artifact (if no other copy is available)
2. Scholarly repositories
 - *explicit deposit* by identified individuals of one or more of the following:
 - software bundles with associated extrinsic metadata
 - extrinsic metadata associated with an artifact already existing in the universal archive
 - *non-public deposits* and/or *embargo periods*
 - *editing* of extrinsic metadata
 - (optional) *moderation* of extrinsic metadata
 - *download* of the deposited bundle (as-is) and the associated metadata

Component Interactions

1. Repositories MUST feed the universal archive
 - all public explicit deposits are integrated in the universal archive
 - software bundles and/or extrinsic metadata sent to the archive
 - in case of bundles extracted from a VCS, archival of the full VCS in the universal archive should be triggered
 - reference identifier is returned to the repository, that exposes it

2. Repositories SHOULD keep a local copy
 - may be mandated by institutional, national, or regional policies
 - download more efficient via the repository than via archive export
3. Universal archive MUST keep track of the origin of the deposit
4. Universal archive MUST provide provenance information to the repository
 - support disambiguation of repository deposits

Long term preservation of the Universal Software Archive

While the main focus of this section is to detail the interconnection with scholarly infrastructures, it is important to also address the issue of long-term preservation for the Universal Software Archive: the best way to guarantee it is by its replication and diversification through a geographically distributed network of mirrors, implemented using a variety of storage technologies, controlled by different institutions. Mirrors must preserve source and all related information: the development history, their revisions, which carry precious insights into the structure of programs and track inter-project relationships. A reliable network of mirrors of the Universal Software Archive built by Software Heritage represents therefore a fundamental component of the infrastructure architecture which must be implemented from the very beginning.

4.1.2 Reference

Objective

Support reproducibility and verifiability of research results:

- ensure unambiguous identification of one or more of the following:
 - a software artifact, optionally in its context
 - the associated metadata

Components

1. Intrinsic identifiers

Specifically designed for software source code, minimal trusted base (only the algorithm needs to be agreed upon)

 - decentralized, independent *identification of all software artifacts*, including files, directories, commits, releases, tags and snapshots
 - decentralized, independent *verification* of the associated software artifacts:
 - technical impossibility to change the object associated with an identifier independently of administrative processes (cryptographically strong hashes)
 - built-in identification of *duplicates*
 - *compatibility* with broadly accepted *industry standards*
2. Extrinsic identifiers

Register based, require structured administrative oversight

 - repository controlled *identification of explicit deposits* and their *associated metadata*
 - identification of *non-digitally native information*, in particular:
 - the notion of a *software project*, as opposed to a specific *software artifact*

- *editing of the metadata associated with a deposit without changing the identifier compatibility with the traditional workflow in scholarly ecosystem*

Trust Model

It is important to notice that intrinsic and extrinsic identifiers have very different characteristics when it comes to the trust model involved. Extrinsic systems of identifiers require an infrastructure that supports the operations related to the registry that contains the association between the identifier and the (metadata that describes the) object that it designates. The persistence and faithfulness of the association of an identifier with a designated object depends on third parties that need to be trusted⁴⁰, as shown by (a) and (b) of Figure 4 that we reproduce here from Section VI of (Di Cosmo, Gruenpeter, & Zacchiroli, 2020). Intrinsic identifiers do not need such an infrastructure at all: the only trusted component is the algorithm that is used to compute the identifier from the object itself. This is one of the main reasons why one should never replace an intrinsic identifier with an extrinsic identifier, when both are available for a digital object.

Requirements

- All references to a publicly available software artifact MUST include a qualified intrinsic identifier; references to a non-publicly available software artifact SHOULD include an intrinsic identifier.
- References to research software artifacts that are explicitly deposited in a scholarly repository MUST include the corresponding extrinsic identifier.
- References to software projects that are not software artifacts MUST include a qualified extrinsic identifier.

Recommendations on Identifier Systems

- Use formally specified, open, non-proprietary, version control independent intrinsic identifiers: SWHIDs⁴¹ are recommended.
- Use formally specified, open, persistent, non-proprietary extrinsic identifiers.

The joint FORCE11/RDA Software Source Code Identification WG has recently released a comprehensive report that details the various use cases and existing approaches for identifying software source code (Gruenpeter et al., 2020). DOIs have a distinct advantage among the various systems of extrinsic identifiers because they have a critical level of adoption in the scholarly publishing world. We recommend that an inclusive approach is explored to guarantee that existing well-established extrinsic identifiers are taken into

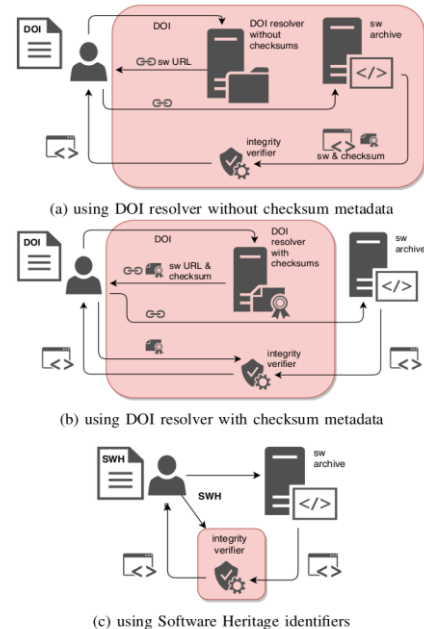


Figure 4. Trusted third parties (shown as rounded red boxes) for software artifact retrieval and verification in three different

⁴⁰ This fact is clearly stated, for example, in the specification document of the Handle system, of which DOI is an instance: "The only operational connection between a handle and the entity it names is maintained within the Handle System. This of course does not guarantee persistence, which is a function of administrative care." (Sun et al., 2003)

⁴¹ <https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html>

account, and we refer to the "PID Architecture for the EOSC" document for further details of implementation in the EOSC (Schwardmann et al., 2020)⁴².

4.1.3 Describe

Objective

Support *discoverability* of (research) software artifacts

Components

1. Metadata
 - **intrinsic:** found in the source code itself
 - **extrinsic:** created via a deposit, publication, or aggregation process
2. Vocabularies and ontologies
3. Tools to create, edit, validate, and convert metadata
4. Registries to store metadata

Requirements

In order to support interoperability:

1. Metadata **MUST** be made available in a *machine-readable form* using a *standard vocabulary adapted for software*. CodeMeta (Jones et al., 2016) is a good candidate for the following reasons:
 - extension of the schema.org⁴³ standard
 - extensive vocabulary designed to allow mapping to other metadata vocabularies⁴⁴ (including CFF⁴⁵ and many others)
 - embryonic community process to extend it
2. Intrinsic metadata **MUST** be created and stored according to recognised best practices in software development⁴⁶.
3. Metadata **SHOULD** support relations:
 - versioning (part of same software, new version, etc)
 - relations with other research objects (papers, etc)
 - relations with other identifiers (DOI vs SWHID)
4. Information specific to a software artifact **SHOULD** be in the intrinsic metadata
 - ensures authors/developers maintain it
 - reduces metadata entry effort and copy and paste errors when a deposit is made

⁴² Bibliography entry will be added when the report on "PID Architecture for the EOSC" is published.

⁴³ <http://schema.org/>

⁴⁴ <https://codemeta.github.io/crosswalk/>

⁴⁵ <https://citation-file-format.github.io/>

⁴⁶ See for example <https://reuse.software/> and <https://www.tldp.org/HOWTO/Software-Release-Practice-HOWTO/>.

Recommendations

- Publishers **MUST** ensure that software associated with the publication is equipped with proper metadata
- Scholarly repositories **SHOULD** provide the necessary means to support metadata curation

4.1.4 Cite/Credit

Objective

Give credit to the authors of research software, as advocated in (Smith et al., 2016), supporting an objective and quality assessment of individual contributions, and improving the current practice, that is far from being satisfactory (Howison & Bullard, 2015; Pan et al., 2019; Zhao & Wei, 2017).

Components

1. Classification of contributor roles for research software

A detailed proposal based on a decade long experience at INRIA and CNRS in France is available in (Alliez et al., 2020), where the following roles are identified:

- architecture
- coding
- debugging
- design
- documentation
- maintenance
- management
- support
- testing

2. Bibliographic citation *data model* adapted for software

This is *the subset of software metadata* needed for producing a citation in all contexts of interest. A proposal from the Inria software citation working group is available at <https://gitlab.inria.fr/gt-sw-citation/bibtex-sw-entry/-/blob/master/swentry.org>. A link to an external source should be used for the rest of the software related metadata (e.g. the affiliation, address, and roles of the authors need not be part of the citation data model). This link can be a PID pointing to a record in a registry (for example, a DOI giving access to the extensive DataCite metadata collection (DataCite Metadata Working Group, 2017), an institutional repository identifier like the one provided by HAL (Di Cosmo, Gruenpeter, Marmol, et al., 2020), or a discipline-specific identifier like the ones that have been used for astrophysics software for decades (Allen & Schmidt, 2015)

3. *Machine readable representation* of the data model

A proposal from the Inria software citation working group for the Bib(La)TeX data format is available at <https://gitlab.inria.fr/gt-sw-citation/bibtex-sw-entry/-/blob/master/swentry.org>.

4. Citation styles for typesetting the citation data

A full-fledged citation style specifically designed for software is available on CTAN in the `biblatex-software` package⁴⁷ (Di Cosmo, 2020a); it is implemented in a modular way, so it can *add support for software citations to any of the other hundreds of citation styles* available for BibLaTeX users. This reference implementation of a software citation style may be used as a touchstone for adding support for software in existing citation styles.

5. Plagiarism detection mechanisms

As research software becomes an evaluated item, plagiarism is inevitably bound to emerge, and proper tools to detect it are needed. Similar tools are used in industry for *license compliance*, a very expensive process. Lighter tools can be provided by Software Heritage. As a minimum, standard manual checks should be done by the publishers

6. (optional) Expert peer evaluation

Several levels of peer evaluation can be implemented:

- **best:** a process that ensures *sufficient quality* of research software *forming an integral part of an accepted scientific publication* (see the Artifact Evaluation process popular⁴⁸ in Computer Science conferences, as published for example in the Dagstuhl Artifacts series (DARTS)⁴⁹, the ACM Badging schema⁵⁰, and the practice of the IPOL Journal⁵¹)
- **good:** a process that ensures that research software is *novel*, developed, and documented properly, and works as expected (see e.g. the criteria of the Journal of Open Source Software⁵²)
- **medium:** a process that ensures that research software can be actually installed and used to produce the results published in a research article (see e.g. the Reproducibility Label⁵³, and the CODECHECK proposal⁵⁴)
- **minimal:** a process that ensures software is properly archived and well referenced in the publication without any review of the source code

4.1.5 Easing Adoption

A few guiding principles apply across all the architecture to ease adoption and improve data quality:

- *added value for the researcher:* any task required of the researcher should provide immediate value for the researcher themselves
 - e.g.: automatic generation of bibliographic entries, curricula, or form filling
 - e.g.: transparent and verifiable metrics (downloads, views), possibly aggregated across repositories
- *metadata validation:* metadata that may be used for credit or evaluation should undergo human validation

47 <https://www.ctan.org/tex-archive/macros/latex/contrib/biblatex-contrib/biblatex-software>

48 <https://www.artifact-eval.org/about.html>

49 <https://www.dagstuhl.de/publikationen/darts/>

50 <https://www.acm.org/publications/policies/artifact-review-badging>

51 <https://www.ipol.im/>

52 https://joss.readthedocs.io/en/latest/review_criteria.html

53 <https://rrpr2018.sciencesconf.org/resource/page/id/5>

54 <https://codecheck.org.uk/process/>

- *avoid duplication*: the same information should not be entered in different formats/places, and in particular:
 - users should not be required to manually enter information that can be extracted from machine readable metadata
- *actionable guidelines*: guidelines should provide easy to follow and implementable steps
- *favour automation and integration among infrastructures*

4.2 Exemplarity Criteria for Participating Infrastructures

Funding agencies and public bodies are looking at exemplarity criteria for funding open scholarly infrastructures (Bilder et al., 2015), and have started to roll out guidelines on how funding decisions related to scholarly infrastructure should be made. In the Open Access area, for example, one can find general principles shared among organizations like COAR and SPARC (COAR & SPARC, 2019), GO-FAIR, or the French NFSO (French National Committee for Open Science, 2019).

Many of these criteria concern the *openness* and *availability* of the *metadata* associated with the scholarly output: these are clearly relevant to our setting, and we incorporated them in the design of the architecture presented in this section. Other criteria concern the openness, sustainability, transparency, and governance of the infrastructures themselves.

We sum up here the criteria that the WG believes are of particular importance for scholarly infrastructures that handle software of interest for research.

Openness

- metadata should be accessible in a standard format and under a CC0 license
- access to the metadata and the data should be possible through an open API using standard protocols and without identification
- aggregated metadata should be available “as open as possible as closed as necessary” (e.g. to respect GDPR regulations)
- the infrastructures should be built from stable existing open source software building blocks, and all the software of the infrastructure should be available under an open source license
- communications and data exchange use open standards for data formats and protocols
- the infrastructure should be hosted and run by a non-profit organization to avoid risk of proprietarisation

Governance

- clear definition of governance bodies
- procedures for the selection of governance bodies’ members are clearly and publicly stated
- procedures for participation are clearly and publicly stated

Sustainability

- the general operation of the infrastructure or platform is not based on the financing of one-off projects
- a plan for long term availability of the service exists and is made public
- an exit strategy that could give continuity to the data and metadata beyond the life of the service

Transparency

- terms of use are clearly and publicly stated
- sources of funding are clearly and publicly stated

4.2.1 Accommodating Innovation

New innovative services may appear in the future, and one of the goals of the architecture proposed here is to make it possible for these innovations to be included easily.

This is made possible by the fact that the proposed architecture relies on *open standards* and *open formats* for the communication and exchange of information between the various components. In particular, we are proposing to adopt a common *open standard* for exchanging *metadata* (CodeMeta), a common *open standard* for intrinsic identifiers (SWHID), to recommend the use of *open APIs*, and the availability of the source code of the infrastructures themselves as *open source*.

This way, if new innovative services for archival, publishing, or aggregation emerge, they will be able to interoperate seamlessly with the other existing components of the architecture.

4.3 Possible Workflows

In this section, we provide more details about reference scenarios that have been identified, together with the corresponding sequence diagrams that allow visualisation of the steps involved.

In the following, we distinguish the following roles:

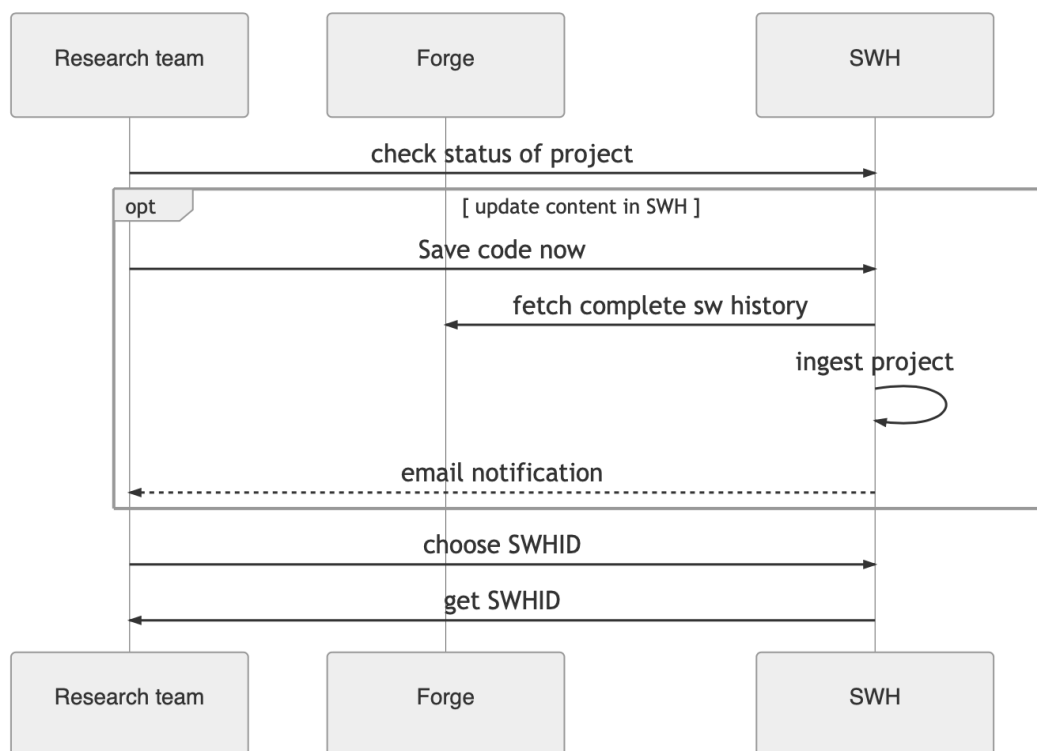
- **research team:** develops or users of software source code; interested in ARDC
- **forge:** code hosting platform used for collaborative development of the source code
- **publisher:** academic publishing entity (article and/or source code review)
- **scholarly repository:** repository run by a research institution (e.g. HAL, Zenodo)
- **catalog/aggregator:** (e.g. OpenAIRE, swMATH, Scanr)
- **SWH:** The Software Heritage universal source code archive
- **data source:** source of input (publications, metadata) for aggregators

4.3.1 Self-Archiving

A research team self-archives a software artifact. As already described in the Archive section⁵⁵, this can be done directly in SWH, or through an Scholarly Repository, which may or may not support a certain level of quality review/curation of the *metadata* of the source code.

Archive from Forges in SWH (Manual and Automated)

The simplest variant for teams that develop their software using publicly accessible code hosting platforms. The research team checks whether (the latest version of) its software project is already archived in SWH, requests its archival if needed, and then gets the corresponding SWHID. The request for archival may be submitted manually, or automatically, as part of a release process, or continuous integration process, via the SWH API. This corresponds to the workflow described in detail in (Di Cosmo, 2020b).



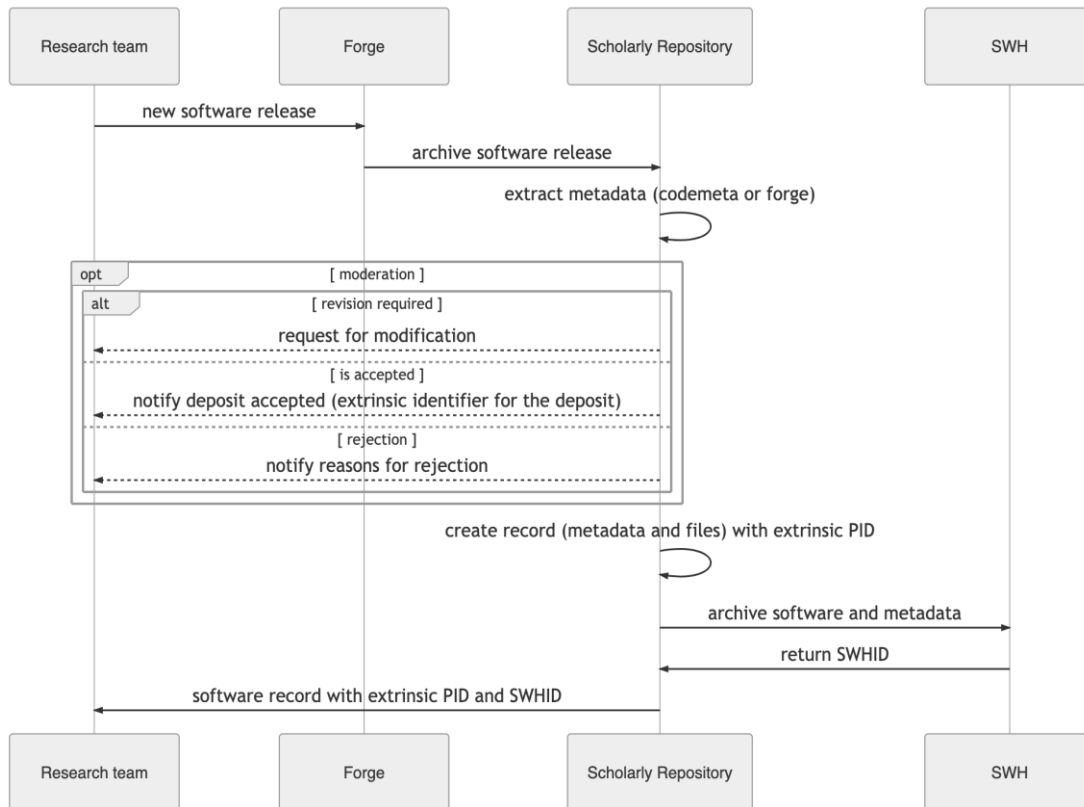
The workflow described in the above diagram can be applied not only to a specific software project developed by a particular research team, but to *any publicly available software project* on a code hosting platform.

Automated deposit of New Releases into Scholarly Repository and SWH (Manual and Automated)

If the research team has a clear release process in place, and has chosen a designated scholarly repository, it may be possible to automate the process of deposit in a scholarly

⁵⁵ <https://hackmd.io/LmSc9a3rRUWkYWQx5MIo9g#markdown-header-archive>

repository, which may trigger a new moderation of the updated content (like what happens with the deposit of new versions of a research article in ArXiv or HAL).



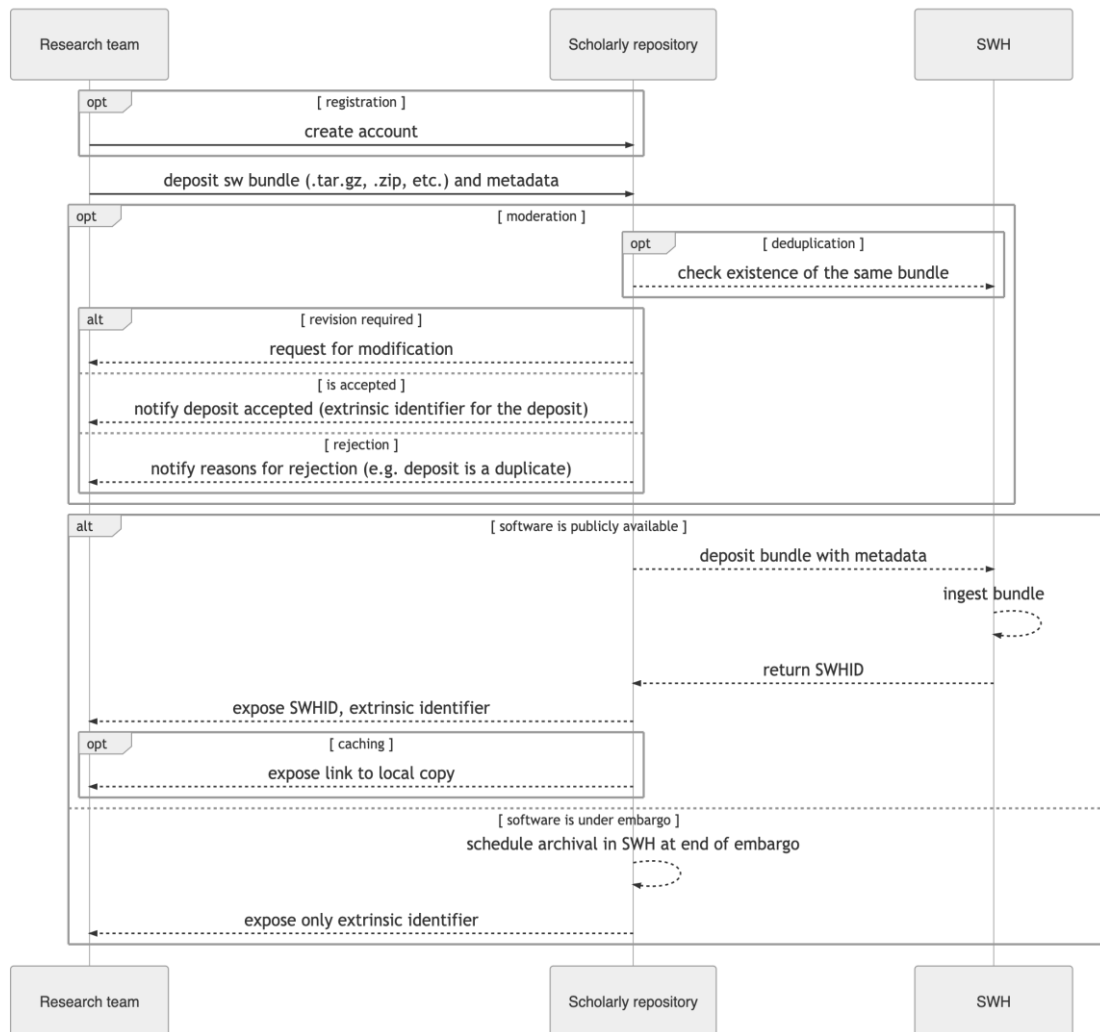
The workflow above, except for the moderation and archival in SWH, has been implemented to connect the GitHub forge with Zenodo.

Deposit Bundle Through an Scholarly Repository

A research team explicitly deposits a *software bundle* (.tar.gz, .zip, etc.) and associated *metadata* into a scholarly repository. The scholarly repository may implement a *moderation* mechanism to ensure a certain level of *quality* of the deposit (deduplication, affiliations of the team members, coherence of the metadata, etc.). Once accepted, the bundle and metadata are archived in SWH, either immediately, or at the end of the optional embargo period.

The sequence diagram below represents the steps already implemented (except for the optional deduplication) by HAL⁵⁶, the French national open access repository, and described in detail in a dedicated research article (Di Cosmo, Gruenpeter, Marmol, et al., 2020).

⁵⁶ <https://hal.archives-ouvertes.fr/>



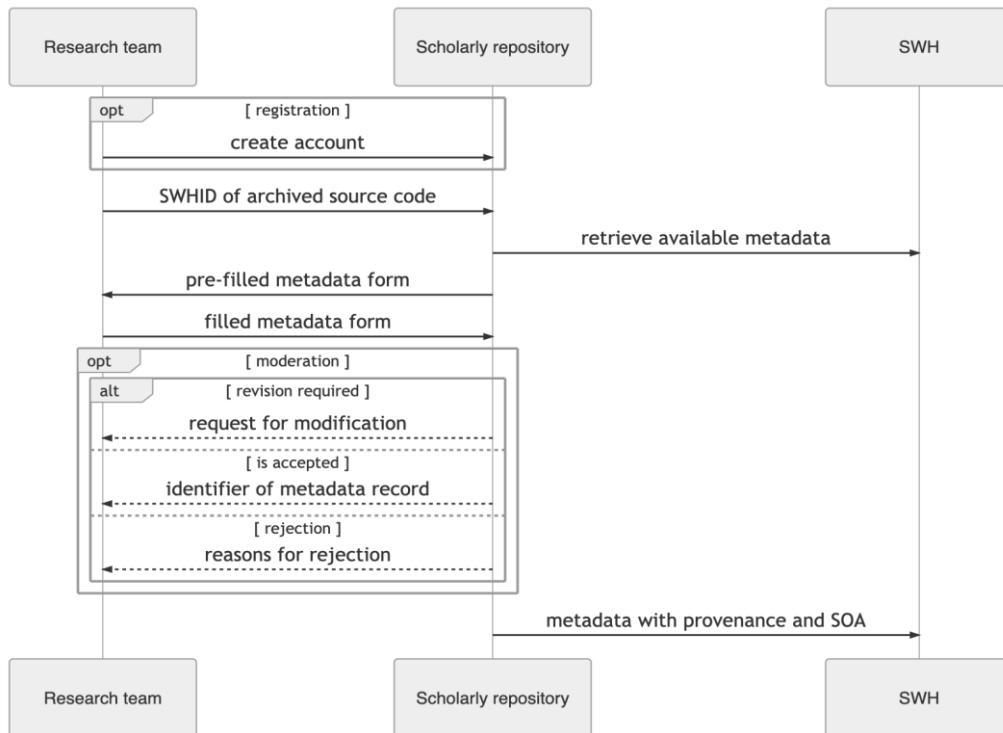
The simplified workflow obtained by removing the moderation step is currently being considered for interconnecting InvenioRDM⁵⁷ with SWH.

Registering Already Archived Software in a Scholarly Repository

A research team may need to register an artifact that has been already archived in SWH in a scholarly repository. To avoid duplication of work, machine readable metadata contained in designated files in the source code should be used to prefill the metadata deposit form. This workflow is currently being implemented in the HAL⁵⁸ open access repository.

⁵⁷ <https://invenio-software.org/products/rdm/>

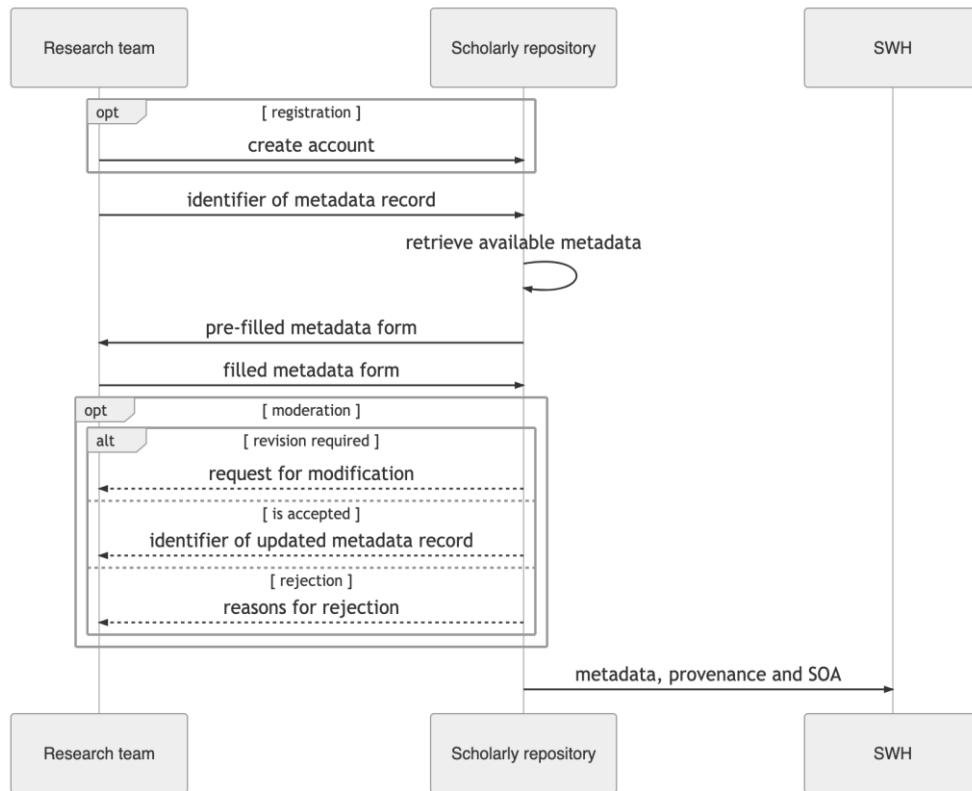
⁵⁸ <https://hal.archives-ouvertes.fr/>



Updating Existing Metadata in a Scholarly Repository

A research team may also need to update the metadata registered in a scholarly repository about an already archived software artifact. This workflow is currently being implemented in the HAL⁵⁹ open access repository. Zenodo also allows the uploaders to update the metadata, and foresees curation features in the future.

59 <https://hal.archives-ouvertes.fr/>



Previous versions of the metadata record should be recorded and available, together with information on who changed what, when, and why.

4.3.2 Scholarly Publication with Associated Source Code

In this scenario, a research team submits an article with associated software source code to a publisher. Several variants are possible, depending on the level of review/curator that the publisher implements. We detail a few of the relevant cases.

Source Code Fully Handled on the Author Side

The publisher does not implement any dedicated workflow for the review/curator of the source code and/or of the associated metadata. In this case, the research team self-archives the relevant source code, following any of the self-archiving workflows described above, and includes the proper identifiers in the final version of the publication.

Publisher Implements Review on Publicly Available Source Code Hosted on Public Forge

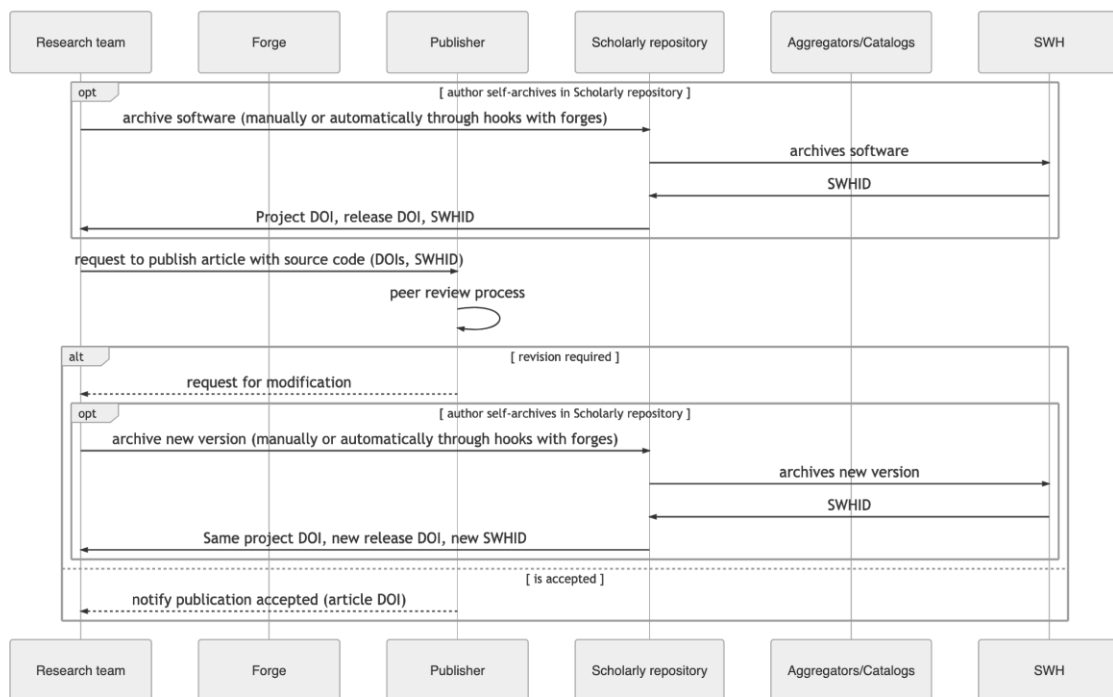
The publisher supports a certain level of quality review/curator of the source code and/or of the associated metadata. The workflow does not detail the review process, which may vary depending on each publisher’s chosen workflow, but we remark that access to unmodified source code is necessary to the review, and for publicly available source code this implies that the identity of the authors of the code is necessarily exposed. This makes it impossible to implement double anonymous review of source code (i.e. the authors do not know the reviewers, and the reviewers do not know the authors). As a consequence, one can find either review processes where the authors are fully known to the reviewers (the most common option in journals, see for example what IPOL⁶⁰ does; it can be open

60 <https://www.ipol.im/>

review or single anonymous review), or a two phase review process, with a first phase focused on the article itself (this may be made double anonymous), and a second phase, once the article is accepted, focused on the software, with the authors of the software known to the reviewers⁶¹ and free to interact with them (this is what is done by the Artifact Evaluation Committees⁶² put in place by tens of prestigious conferences in computer science and recommended in the ACM Badging system⁶³; the Dagstuhl DARTS series⁶⁴ publishes a selection of the results of these processes).

The publisher propagates metadata about the acceptance of the publication containing the software artifact into a Scholarly Repository (which propagates to SWH) or directly to SWH, and optionally to relevant aggregators/catalogues; notice that in general this is extrinsic metadata about the software artifact. The publisher may also play a role in the archival of the source code associated with the published article. A few variants are shown below.

Variant 1.a. Author self-archives in a Scholarly Repository



This workflow shows a common case in scholarly publication, which is the usage of Project DOIs that facilitate the review process by keeping the same PID throughout all the revisions.

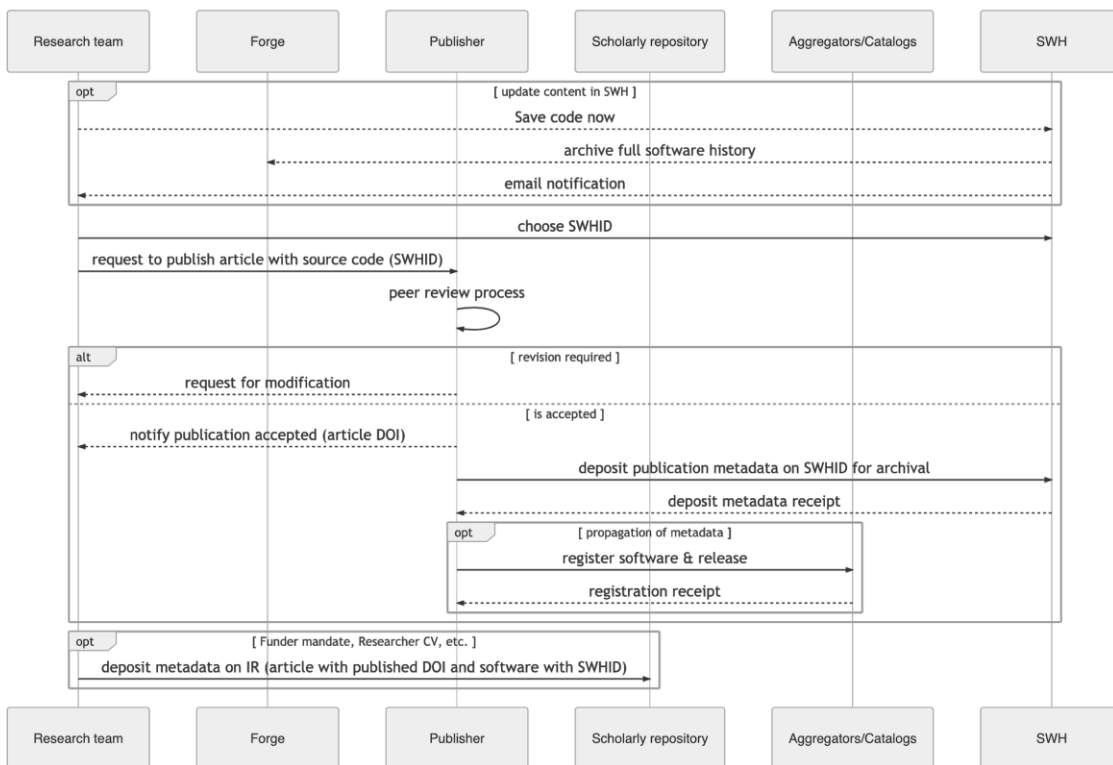
61 We remark here that in all major CS conferences that implement software evaluation, the Artifact Evaluation Committee is separate from the program committee that reviews the articles. The AEC steps in after acceptance of the paper, and can openly interact with the software authors.

62 <https://www.artifact-eval.org/>

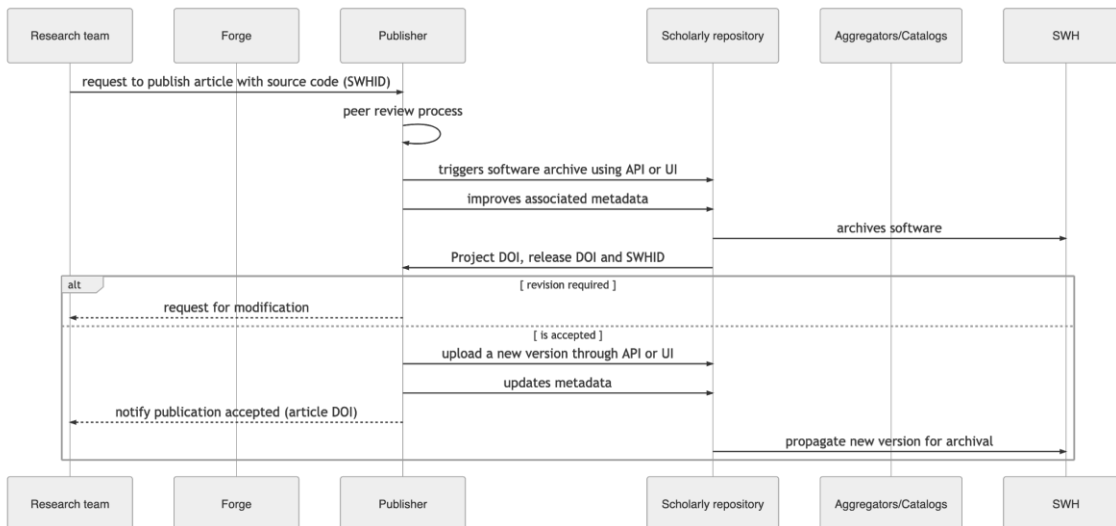
63 <https://www.acm.org/publications/policies/artifact-review-badging>

64 <https://www.dagstuhl.de/publikationen/darts/>

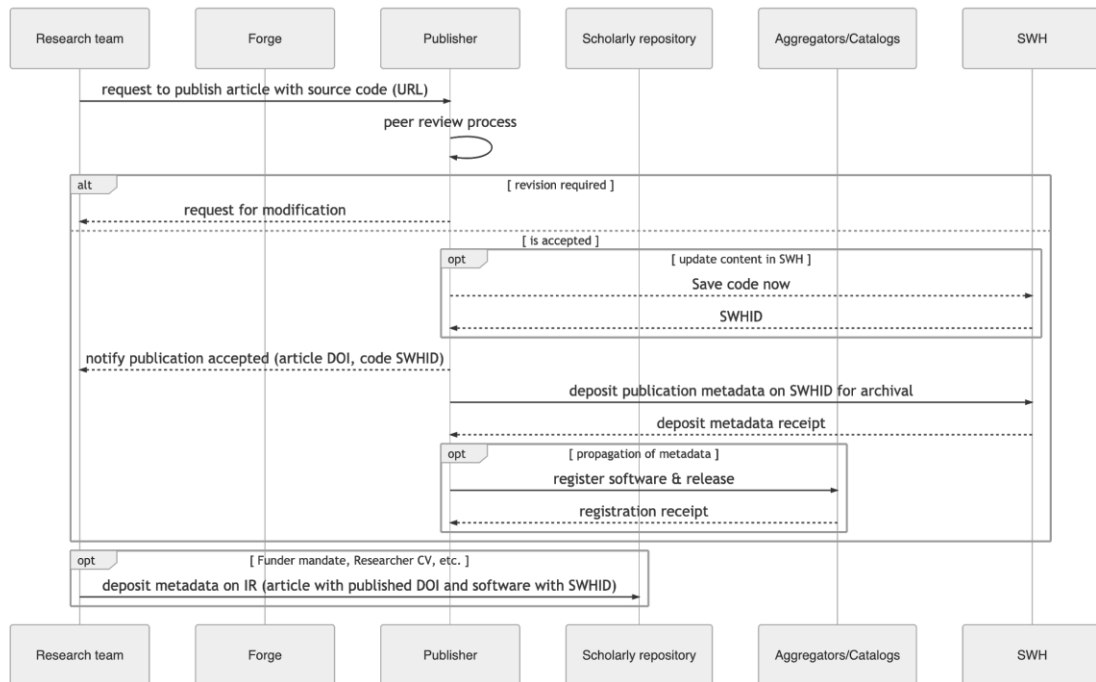
Variant 1.b. Author self-archives directly in SWH



Variant 2.a. Publisher archives in scholarly repository



Variante 2.b. Publisher archives directly in SWH



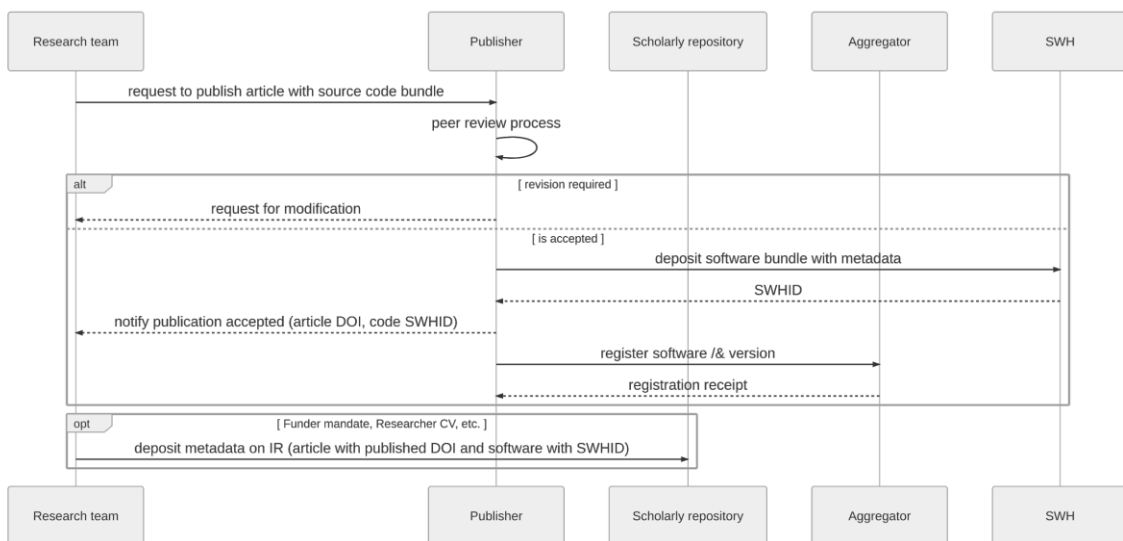
This variant requires less author intervention, as the publisher handles all the archival steps.

Publisher Implements Review on Source Code Submitted as a Bundle

If the software is submitted as a bundle, it needs to be archived through the SWH deposit API, either by the publisher, or via a scholarly repository.

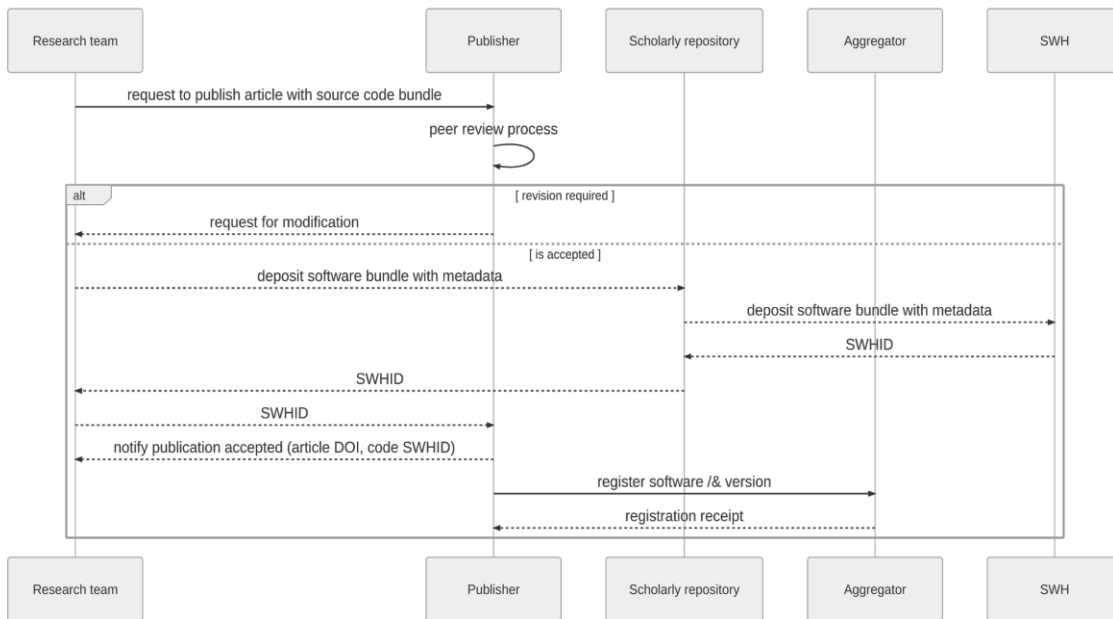
Variante 1. Publisher archives directly in SWH

This is the workflow currently implemented by IPOL⁶⁵.

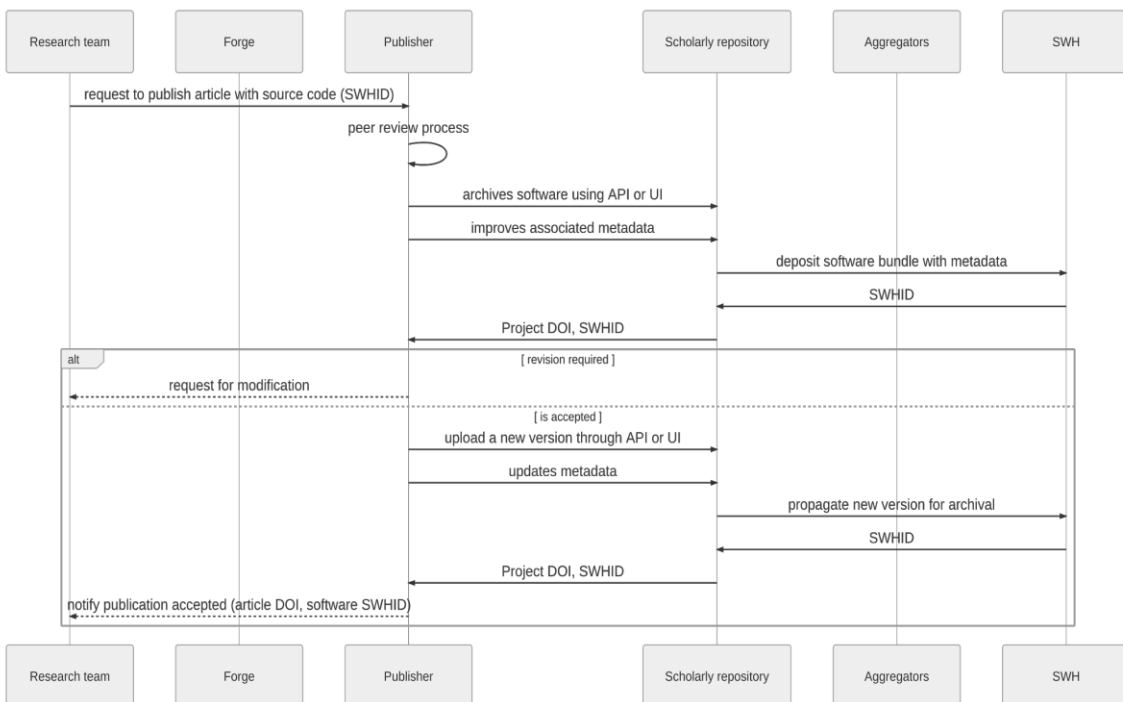


65 See <https://www.softwareheritage.org/2020/06/11/ipol-and-swh/>.

Variant 2. Publisher requests author to archive in scholarly repository



Variant 3. Publisher archives in scholarly repository



A workflow similar to the above is being implemented in a prototype developed by Dryad and Zenodo⁶⁶.

66 See <https://blog.zenodo.org/2020/03/10/dryad-and-zenodo-our-path-ahead/>

Conferences with Artifact Evaluation Committees

In computer science, differently from what happens in many other disciplines, the research community is used to publishing more in conference proceedings than in journals. This has a series of interesting consequences, ranging from the program committee changing regularly, to the fact that in many cases the identity of the conference has little to do with the publisher of the proceedings. Here, we are interested in a workflow based on the one that has been widely adopted since 2011 in many prestigious computer science conferences that have put in place an Artifact Evaluation Committee (AEC) (Childers et al., 2016)⁶⁷.

The workflow shown below differs from the previous ones in several respects:

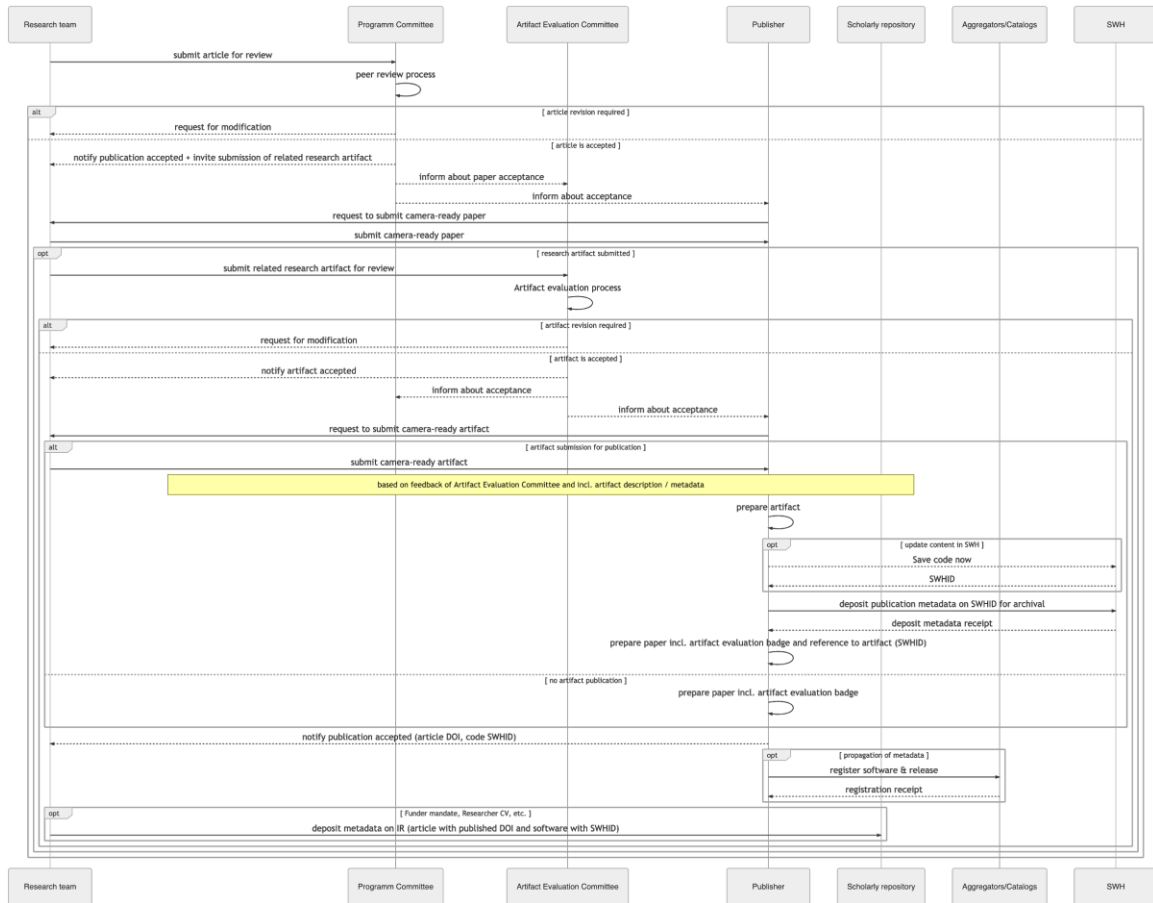
- the Program Committee of the conference is not bound to a particular publisher
- the evaluation of the artifact is performed by a dedicated AEC, *distinct from the Program Committee, only for accepted articles*; as a consequence, anonymous review is no longer needed when it comes to artifacts and the AEC can freely interact with the authors;
- the publisher is not involved at all in the artifact evaluation process. It only comes into play after the artifact has been positively evaluated, to perform two actions
 - save the artifact, obtaining a persistent identifier
 - add a badge to the article, with an optional link to the saved artifact

Currently, support for this process is not fully satisfactory, as the artefact evaluation workflow is not supported natively in the conference handling software, and the archival of the artifact is often left to the authors. Some conferences do recommend archival in SWH⁶⁸, or in the publisher's own digital library, but we believe it would be better to have the publisher take responsibility for ensuring archival, as depicted in the workflow. Proper support for artifact evaluation and archival should be added to the software used in publishing systems, with different levels of quality review. This would allow the uptake of the AEC process, or processes inspired from it, more broadly, in conferences and in journals.

⁶⁷ See the <https://www.artifact-eval.org/> for the seminal idea, and <http://evaluate.inf.usi.ch/> for an actual list of conferences, and updated bibliography. The ACM has adopted a similar badging schema, but does not mandate any particular process for the evaluation, see <https://www.acm.org/publications/policies/artifact-review-badging>

⁶⁸ See for example <https://popl20.sigplan.org/track/POPL-2020-Artifact-Evaluation>

Scholarly Infrastructures for Research Software

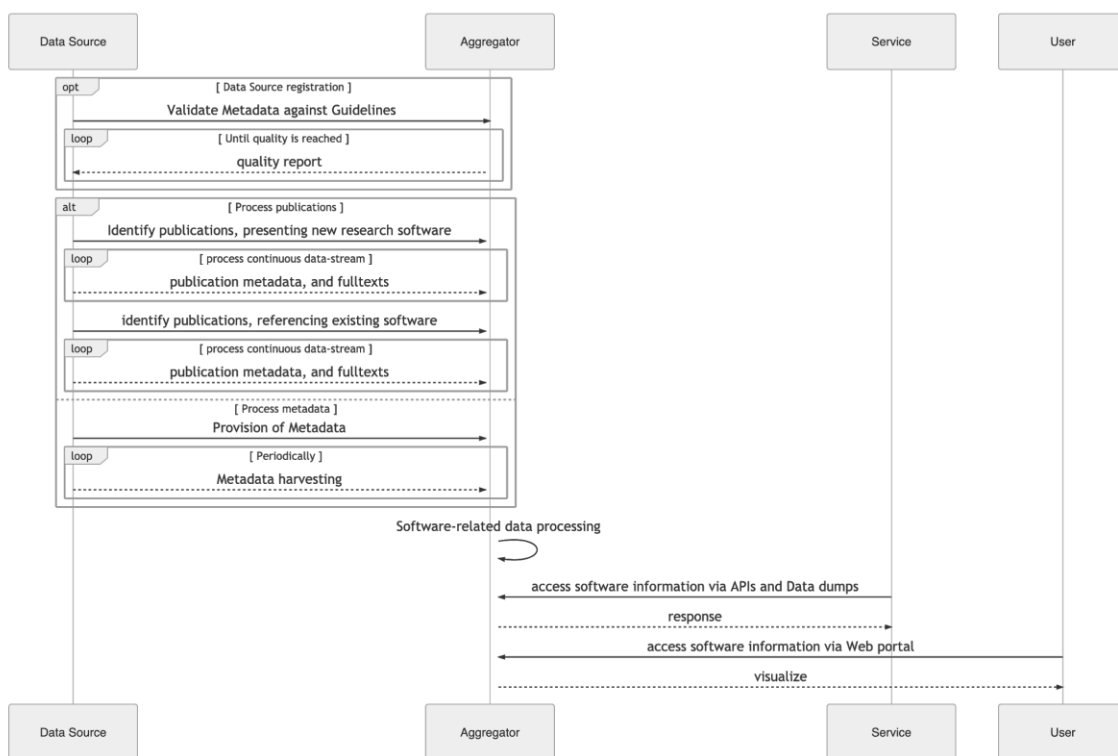


4.3.3 Aggregators

It is being identified that aggregators fit into two groups or models: Push model and Pull model. The former is a model in which *Data Sources* send proactively software-related metadata and publications to the *Aggregators*; the latter is a model in which the *Aggregators* fetch software-related metadata and publications directly from the *Data Sources*. Independently of the model, aggregators main role in the scholarly publishing workflows is always to harvest information, process it (harmonization, deduplication, enrichment, etc) and provide the aggregated outcome to the user community, usually with extra added value coming from such aggregation. The aggregated information can then be consumed again by the very same Data Sources (e.g. Archives) to provide better services to final users.

Pull model

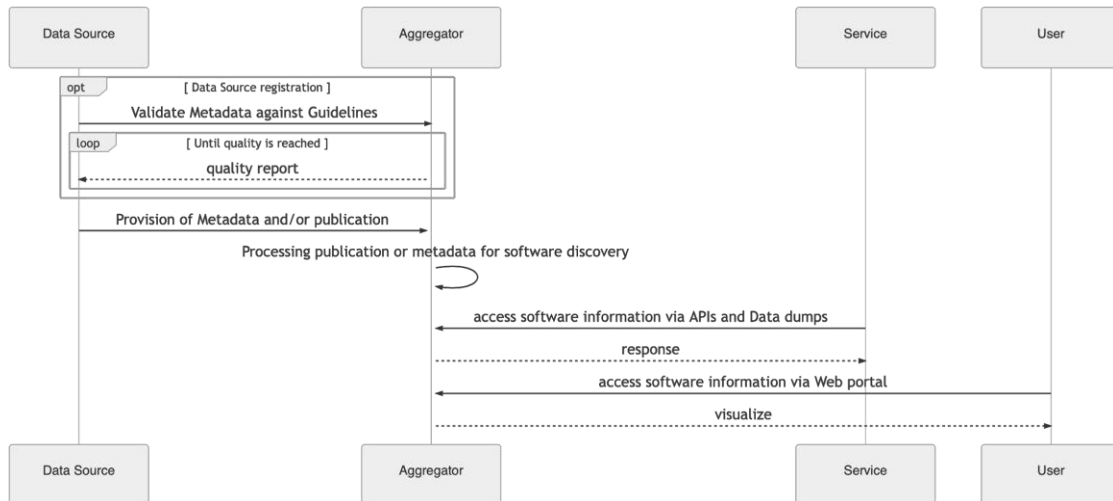
Aggregators proactively harvest software-related metadata and/or publications from Data Sources. In the case of publications, they are processed, software-related information is extracted and uniquely identified. In the case of metadata, it gets validated and merged into the internal graph of information.



Both, *swMATH* and *OpenAIRE* implement most of this workflow, with the difference of focusing on a specific domain (mathematics) or multi-domain.

Push model

Aggregators enable APIs to other Services (e.g. Archives) to allow them to submit new publications (including metadata) directly, without the need of harvesting.



OpenAIRE could also fit in this Push Model, as well as Papers with Code⁶⁹. DataCite, although not a traditional aggregator, could be seen as one and fitting in the Push model; this is because Scholarly Repositories push software-related metadata directly to DataCite and the aggregated metadata is available via their APIs and User Interface.

69 <https://paperswithcode.com>

5 RECOMMENDATIONS

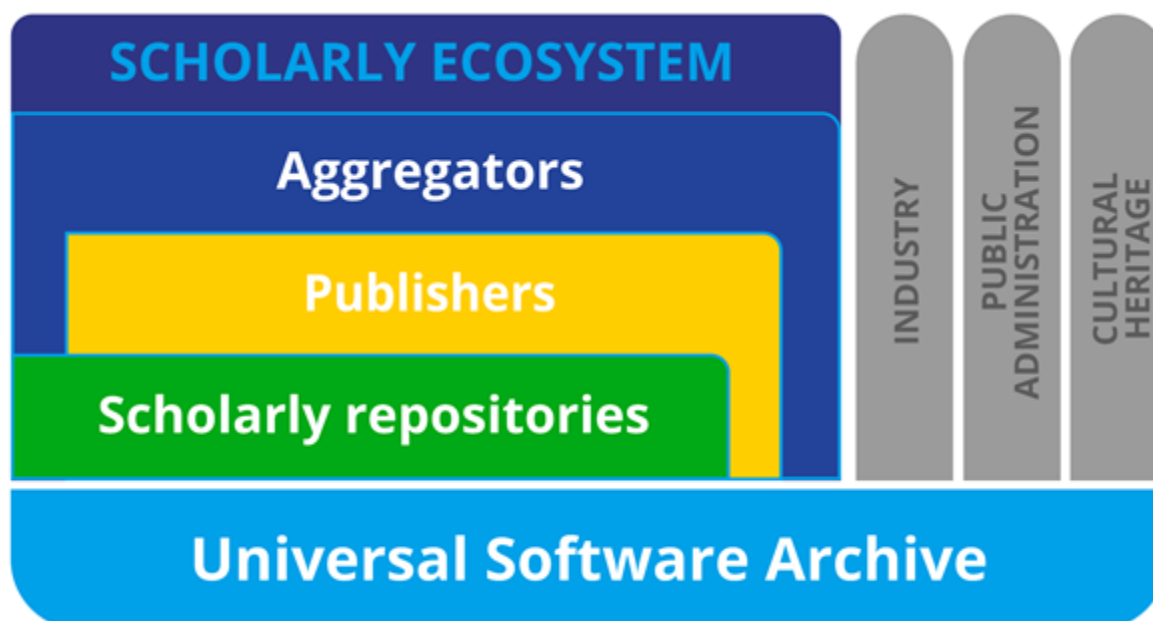


Figure 5. Architecture of interconnected scholarly infrastructures supporting archival, reference, description and citation of (research) software source code.

As recalled in Section 2 Introduction of this report, building a proper architecture of connected scholarly infrastructures for research software is a significant undertaking, and requires *standards, tools, infrastructures, training, outreach, and involvement with the publishing community*. It also needs *proper funding* both for the *development, communication, and outreach efforts*, and for the *operational costs*. In this section, we summarise the key recommendations that emerged from the analysis of the current needs and state of the art, and the design of the future architecture.

5.1 Funding Development of Tools, Standards, and Guidelines

The following set of recommendations includes *actionable items* that should be turned into concrete development projects in the short term.

5.1.1 Interactions

As discussed in Section 4.1.1 Archive, it is important to ensure a vertical interconnection between a universal software archive and scholarly repositories, for the latter to feed the universal archive (see Figure 5). This requires engineering and funding for the development of proper adaptors. Deployment is foreseen in a 2-year timeframe. Simultaneously, it should be avoided that publishers implement their own solutions for software archival. Rather, they should rely on scholarly repositories or a universal software archive to ensure software preservation and citation. In order to include the act of depositing or archiving software source code in scholarly repositories and a universal software archive in the publishing workflow it is necessary to adapt the publishers' internal processes, and we consider that this may be implemented in a 2-year timeframe. Development of tools for automating the software source code archival and reference workflow needs engineering and funding, and can be implemented in a 4-year timeframe. Additionally, it should be ensured that curated metadata is archived alongside the source code archived in a universal software archive to support reproducibility and verifiability of research results.

Therefore, engineering and funding of tools for automating the curated metadata archival workflow is required and is foreseen to be implemented in a 4-year timeframe.

Finally, it is important to ensure a reliable and broad network of mirrors of the Universal Software Archive, taking into account its fundamental importance both for the long-term preservation and to make the full content of the archive readily available for research activities. An appropriate coordination activity of this network of mirrors should also be funded.

5.1.2 *Metadata About Software*

The following set of recommendations concern metadata about software. First, all metadata about software must be licensed as Creative Commons CC0. Second, all metadata exchange between the different components of the architecture should be based on the CodeMeta vocabulary. EU representatives should get involved in the CodeMeta community and help establish a stable, long-term governance. Also, additional converters and adaptors should be developed as needed to consume and expose metadata using the CodeMeta vocabulary. This requires engineering and funding and can be deployed in a 2-year timeframe. Third, metadata should include all information relevant for software source code in the scholarly world, including in particular licence, identifiers, repositories, authors, and funders that supported the software development (e.g. EU or national grants). Last, it must be ensured that mainstream formats used by publishers for citations (e.g. JATS) properly support all metadata items that are relevant for software citations, and there is a JATS4R recommendation for software citations. Here, contribution to the existing standards is the norm in order to extend them as needed.

5.1.3 *Identifiers*

Creation of new systems of identifiers is unwanted and instead reuse should be fostered. Generalizing the use of the following list of identifiers is recommended (see Section 4.1.2 Reference **Error! Reference source not found.**for details):

- SWHID⁷⁰ intrinsic identifiers for publicly available software source code.
- Extrinsic identifiers for research source code explicitly deposited in a scholarly repository.
- Extrinsic identifiers for software projects.

DOIs have a distinct advantage among the various systems of extrinsic identifiers because they have a critical level of adoption in the scholarly publishing world. We recommend that an inclusive approach is explored to guarantee that existing well-established extrinsic identifiers are taken into account, and we refer to the "PID Architecture for the EOSC" document for further details of implementation in the EOSC (Schwardmann et al., 2020)⁷¹.

5.1.4 *Credit*

To ease adoption, development of tools that can produce appropriate citations for research software source code, and enhancement of existing reference management tools to support the same approach, is desired. Specifically, this includes development of and contribution to the needed extensions to the mainstream reference manager tools (Mendeley, Zotero, etc.) to ensure that the underlying data model can accommodate all the specificities of and roles related to software source code, as identified in (Alliez et al., 2020; Di Cosmo, 2020a;

⁷⁰ <https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html>

⁷¹ Bibliography entry will be added when the report on "PID Architecture for the EOSC" is published.

Gruenpeter et al., 2020; Katz et al., 2020). Additionally, the publishers and the research community at large should work together to produce guidelines about how to cite software specifically, agreeing on a common set of citation styles. Also, publishers must ensure all links/mentions to any code not written for the research in hand are treated as proper references in the bibliography, including all associated required metadata. Metrics that cater to the needs of a variety of actors should be explored and common standards to share and reuse them agreed upon. As detailed in Section 3.4.4 Metrics, these metrics should be open, verifiable, and shareable. These metrics should not be reduced to simple numeric indicators, to avoid reproducing in the research software world the negative effect that bibliographic indicators have had in the research publishing world. It is necessary to bring together a broad spectrum of expertise, and include in the conversation representatives of the research community that will be directly impacted by the creation of these metrics. Publishers must ensure that the peer review process also covers software source code, with the level of evaluation most appropriate for their field, as mentioned in Section 4.1.4 Cite/Credit (point 6), and develop a set of common guidelines for moderation and curation protocols. Development of a set of standard tools and workflows should be funded to support and ease adoption of more sophisticated levels of review, like the ones implemented by AECs.

5.1.5 Policy/Guidelines

Building a proper architecture of connected scholarly infrastructures for research software needs guidelines to increase the treatment of software on equal footing with other research outputs. Specifically, Open Science guidelines for researchers should clearly recommend software deposit in trustworthy scholarly repositories and in the universal software archive maintained by Software Heritage to ensure long term preservation. Simultaneously, Open Science guidelines should raise awareness about the existence of modern approaches to software development (including Version Control Systems, continuous integration, etc.), and encourage their use where appropriate. Last, all publishers must be made aware of the importance of source code and data, and make the publication and archival of these artefacts in conjunction with the article publication mandatory.

5.1.6 Easing Adoption

As detailed in Section 4.1.5 Easing Adoption, it is important to ensure particular attention is paid to ease adoption and increase the quality of the (meta)data collected. To this end, the participation of researchers is of paramount importance, and in developing tools and guidelines one needs to ensure that if the researchers are asked to do extra work, there is an *immediate added value for researchers* (e.g. automatic generation of bibliographic entries, curricula, or form fillers, simplification of existing procedures, etc.) and that *researchers should not be required to manually enter information more than once*, or information that can be extracted from machine readable metadata. A particular sensitive factor is the quality of information that may be used for *crediting authors or evaluating researchers*, so metadata that may be used to this end *should undergo human curation* (Alliez et al., 2020).

5.2 Broader Policy Recommendations for the EOSC

We believe that besides the technical development, the technical standards, and general guidelines for the various actors, the EOSC has a key role to play in ensuring that the overall architecture will be built in a way to best cater to the needs of the research community, in the same spirit of the general principles shared about Open Access infrastructures among organizations like COAR and SPARC (COAR & SPARC, 2019), GOFAIR or the French NFSO (French National Committee for Open Science, 2019).

5.2.1 Criteria of Excellence, and Sustainability of the Architecture

The following set of recommendations provide concrete actionable steps for the EOSC to ensure openness, transparency, good governance, and sustainability of the key infrastructures.

First, EOSC should elaborate a set of criteria of excellence for participating infrastructures that incorporate the principles of openness, good governance, and transparency detailed in Section 4.2 Exemplarity Criteria for Participating Infrastructures. The following list of criteria should be included for the alignment with the principle of openness:

- metadata should be accessible in a standard format and under a CC0 license;
- access to the metadata and the data should be possible through an open API using standard protocols and without identification;
- aggregated metadata should be available “as open as possible and as closed as necessary” (e.g. to respect GDPR regulations);
- the infrastructures should be built from stable existing open source software building blocks, and all the software of the infrastructure should be available under an open source license;
- communications and data exchange use open standards for data formats and protocols;
- the infrastructure should be hosted and run by a non-profit organization, to avoid risk of proprietarisation⁷².

Additionally, the following list of criteria of excellence are included for alignment with the principles of transparency and good governance:

- clear definition of governance bodies;
- procedures for the selection of governance bodies’ members are clearly and publicly stated;
- procedures for participation are clearly and publicly stated;
- terms of use are clearly and publicly stated;
- sources of funding are clearly and publicly stated.

Second, the EOSC should actively get involved with key infrastructures to ensure their long term sustainability, and take part in their strategic evolution, avoiding common pitfalls that

⁷² This is also part of the draft recommendations for Open Science released by UNESCO in October 2020, Section II, point (iv) (UNESCO, 2020).

have been identified over the years in the history of essential Open Source projects (Eghbal, 2016) and that are not identical to the concerns related to infrastructures for research data (Rosenthal et al., 2014). This involves several aspects:

- *Technical*: ensure archives adopt **standard practices** for data preservation.
- *Financial/Institutional*: contribute to a **long-term funding plan** and/or a wind down and migration plan for the key infrastructures. In particular, *do not rely on project money for funding the operation of the infrastructures*.
- *Organisational*: ensure the key components of the architecture are run as a *non-profit infrastructure*, and actively **participate in their governance**.

5.3 Longer Term Perspectives

On a longer-term perspective, we believe that the following objectives should be clearly on the roadmap, and that research and development effort should be spent to address them at the 4-7-year horizon.

5.3.1 Advanced Technology Development

Of importance is the development of an advanced search engine for software source code. This search engine should leverage recent results in machine learning⁷³, and go beyond simple text search, integrating scholarly metadata, provenance and dependency information, and software development metrics. Moreover, the network of mirrors of Software Heritage, together with the scholarly ecosystem (repositories, publishers and aggregators) may help develop an emerging domain of research that will lead to the development of advanced tools at the service of the software community as a whole, "Big Code". This research area leverages new methodologies of artificial intelligence and big data to analyse the entire body of publicly available source code and take full advantage of the knowledge that is sealed within it.⁷⁴

Another longer-term need is the development of an efficient and open plagiarism detection technology on top of the universal source code archive provided by Software Heritage. This will allow archives, publishers, and aggregators to spot near-duplicates and avoid fraud, much like it happens with traditional publications, but without the limitations of the closed datasets and commercial agreements that are needed for articles but do not exist in the open source software world. Similar technology is in use in industry, with high costs and different main objectives, and needs to be retargeted and adapted for the scholarly world. The global corpus of software source code amassed by Software Heritage, together with the global graph of software development that it maintains, is a key enabler for this task. The same plagiarism detection building blocks can be used to trace how a particular research software evolves over time, through forks or other means, and how it can be reused elsewhere.

Moreover, building efficient and open spam filtering tools, which allow filtering out of non-software projects, protects scholarly repositories and archives that do not enforce human moderation of deposits, and eases the work of moderators for those that do.

⁷³ See (Feng et al., 2020) for an example of the many possible features.

⁷⁴ Two mirrors of the Software Heritage archive are currently being developed: one by the *Stockholm* company FOSSID, a leader in open source software compliance and security, and one by ENEA, the Italian National Agency for New Technologies, Energy and Sustainable Economic Development in collaboration in collaboration with the *Department* of Computer Science and Engineering of *Bologna University*. ENEA Mirror will be part of the *Bologna Big Data Technopole*, one of the leading centers for scientific calculation at the European and world level, that will host the ECMWF Data Center and one of the pre-exascale computers financed by EuroHPC Joint Undertaken initiative.

Last but not least, in order to foster full reproducibility of research results, mechanisms, technology, and tools should be explored to ensure that a given executable or a full software system and workflows can be reliably executed again, with proper integration between articles, data, and software. This is a complex subject, with a broad variety of tools and approaches that will need to be surveyed and assessed. The ultimate goal of research and development efforts in this area should be the inclusion of execution environments for research software into infrastructure services available to all researchers, both for performing research and for evaluating submitted or published research.

5.3.2 Policy

While we subscribe to the general statement that all research output should be “as open as possible, as closed as necessary”, we believe that to fully achieve the potential of Open Science, ***all research software should be made available under an Open Source license by default, and all deviations from this default practice should be properly motivated***. We recommend including this clause in all future research funding programs.

6 ANNEXES

6.1 Glossary

Term	Definition
Aggregator	One of the tree typologies of operational infrastructures this report is targeting. Any service that collects information about digital content from a variety of sources with the primary goal of increasing its discoverability, and possibly adding value to this information via processes like curation, abstraction, and classification, and linking. This class of service, that include scholarly catalogues and indexes, usually provide a search engine that gives access to a description of the aggregated content, and may provide links to versions of it archived elsewhere. These services may be generalist, or have a disciplinary, geographic or institutional scope
Architecture	The term architecture usually refers to the high-level design of the components needed to build a system, and their relationships. In this report we use the term scholarly architecture of infrastructures to designate the high-level organization and relationship of operational infrastructures that may satisfy the ARDC needs in the scholarly world
Archive	One of the tree typologies of operational infrastructures this report is targeting. Any service that has as one of its primary goals the long-term preservation of the digital content that it collects. This includes a broad spectrum of services, ranging from institutional repositories to disciplinary repositories in the scholarly world, as well as services that have a broader scope than the scholarly world
AEC	Artifact Evaluation Committee. A panel of reviewers, usually disjoint from the program committee, that evaluates the quality of the software artifact associated with a publication accepted in a conference. See https://www.artifact-eval.org/ for more details.
ARDC	An acronym that stands for <i>Archive, Reference, Describe</i> and <i>Cite</i> , i.e. the four pillars that scholarly infrastructures should support for software source code management in the world of research
CodeMeta	A project called to develop a concept vocabulary (the CodeMeta schema) that can be used to standardize the exchange of software metadata across repositories and organizations.
Extrinsic identifiers	Systems of identifiers that rely on a register to keep the correspondence between the identifier and the designated object. Very well-known examples in the scholarly world are ARK, Handle and DOI.
FAIR	A set of principles developed to promote Findability, Accessibility, Interoperability, and Reuse of digital assets, mainly datasets
Infrastructure	See Operational Infrastructure

Term	Definition
Intrinsic identifiers	Identifiers that can be computed from the designated object itself without needing a register to maintain the correspondence. Well known systems of intrinsic identifiers before the digital age are the musical notation and the chemical notation. For software source code, the current standard is the SWHID.
JATS	Journal Article Tag Suite
MUST	Used as in RFC 2119: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification
<i>Operational Infrastructure</i>	A complex system consisting of deployed facilities, equipment, services, policies, procedures and human resources needed for the operation of an organization.
PID	PID stands for "Persistent Identifier", a term generally used to designate systems of extrinsic identifiers (e.g. DOI, Ark, Handle), for which organizational support has been set up to maintain the association between identifiers and the designated objects.
Publisher	One of the tree typologies of operational infrastructures this report is targeting. <i>Any organization that prepares submitted research texts, possibly with associated source code and data, to produce a publication and manage its dissemination, promotion, and archival process. Software and data can be part of the main publication, or assets given as supplementary materials depending on the policy of the journal. In addition, publishers implement a process for ensuring the quality of the accepted research material (usually peer review), which is carried out by the subject-specific community of experts</i>
Research Software	<i>Software that researchers in any discipline may feel the need to have scholarly infrastructure support for, no matter if it is considered a tool, a result or an object of study</i>
Scholarly Infrastructure	An operational infrastructure called to support the scholarly communication process
Scholarly Repository	An organisation called to archive and make available research artifacts, e.g. articles, datasets, software. Examples include HAL, Zenodo, figshare, Dryad
SHOULD	Used as in RFC 2129: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course
SIRS	Scholarly Infrastructures for Research Software
Software Source Code	As very concisely stated in the General Public Licence, "The source code for a work means the preferred form of the work for making modifications to it." This definition includes the common case of human readable instructions usually written

Term	Definition
	as plain text.
SWHID	SoftWare Heritage persistent IDentifier
Universal Software Archive	An organization that maintains an archive that collects, preserves and gives access to all the publicly available software source code, independently of where, why and how it is developed. Currently, this is the role of the Software Heritage Foundation.
VCS	Version control system: software tool set used by developers to track and manage changes made to source code over time.

6.2 Bibliography

- Abelson, H., & Sussman, G. J. S. with J. (1985). *Structure and Interpretation of Computer Programs*. The MIT Press.
- Abramatic, J.-F., Di Cosmo, R., & Zacchiroli, S. (2018). Building the universal archive of source code. *Communications of the ACM*, 61(10), 29–31. <https://doi.org/10.1145/3183558>
- ACM. (n.d.). *Data & software artifacts*. <https://www.acm.org/publications/artifacts>
- ACM. (2016). *Artifact review and badging*. <https://www.acm.org/publications/policies/artifact-review-and-badging>
- Allen, A., Aragon, C. R., Becker, C., Carver, J., Chis, A., Combemale, B., Croucher, M., Crowston, K., Garijo, D., Gehani, A., Goble, C. A., Haines, R., Hirschfeld, R., Howison, J., Huff, K. D., Jay, C., Katz, D. S., Kirchner, C., Kuksenok, K., ... Vinju, J. J. (2017). Engineering Academic Software (Dagstuhl Perspectives Workshop 16252). *Dagstuhl Manifestos*, 6(1), 1–20. <https://doi.org/10.4230/DagMan.6.1.1>
- Allen, A., & Schmidt, J. (2015). Looking before leaping: Creating a software registry. *Journal of Open Research Software*, 3(e15). <http://dx.doi.org/10.5334/jors.bv>
- Alliez, P., Di Cosmo, R., Guedj, B., Girault, A., Hacid, M.-S., Legrand, A., & Rougier, N. (2020). Attributing and Referencing (Research) Software: Best Practices and Outlook From Inria. *Computing in Science and Engineering*, 22(1), 39–52. <https://doi.org/10.1109/MCSE.2019.2949413>
- Arévalo, M., Escobar, C., Monasse, P., Monzón, N., & Colom, M. (2017). The IPOL Demo System: A Scalable Architecture of Microservices for Reproducible Research. In B. Kerautret, M. Colom, & P. Monasse (Eds.), *Reproducible Research in Pattern Recognition* (Vol. 10214, pp. 3–16). Springer International Publishing. https://doi.org/10.1007/978-3-319-56414-2_1
- Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604), 452–454. <https://doi.org/10.1038/533452a>
- Barborini, Y., Di Cosmo, R., Dumont, A. R., Gruenpeter, M., Marmol, B. P., Monteil, A., Sadowska, J., & Zacchiroli, S. (2018). *The creation of a new type of scientific deposit:*

Software. <https://www.softwareheritage.org/wp-content/uploads/2020/01/barborini-rda-poster.pdf> <https://hal.inria.fr/hal-01738741>

Barnes, N. (2010). *Science code manifesto*. <http://sciencecodemanifesto.org>

Benureau, F. C. Y., & Rougier, N. P. (2018). Re-run, Repeat, Reproduce, Reuse, Replicate: Transforming Code into Scientific Contributions. *Frontiers in Neuroinformatics*, 11. <https://doi.org/10.3389/fninf.2017.00069>

Bilder, G., Lin, J., & Neylon, C. (2015). *Principles for open scholarly infrastructures-v1*. <https://doi.org/10.6084/M9.FIGSHARE.1314859>

Bönisch, S., Brickenstein, M., Greuel, G.-M., & Sperber, W. (2012). SwMATH – citations for your mathematical software. *JournalId:00006143*, 2012. <https://doi.org/10.1007/BF03345852>

Borgman, C. L., Wallis, J. C., & Mayernik, M. S. (2012). Who's Got the Data? Interdependencies in Science and Technology Collaborations. *Computer Supported Cooperative Work*, 21(6), 485–523. <https://doi.org/10.1007/s10606-012-9169-z>

Bradner, S. O. (1997). *Key words for use in RFCs to indicate requirement levels*. 2119. <https://doi.org/10.17487/RFC2119>

Buckheit, J. B., & Donoho, D. L. (1995). WaveLab and Reproducible Research. In A. Antoniadis & G. Oppenheim (Eds.), *Wavelets and Statistics* (Vol. 103, pp. 55–81). Springer New York. https://doi.org/10.1007/978-1-4612-2544-7_5

Burton, A., Fenner, M., Haak, W., & Manghi, P. (2017). *Scholix metadata schema for exchange of scholarly communication links*. <https://doi.org/10.5281/zenodo.1120265>

Bussi, L., Di Cosmo, R., Montangero, C., & Scatena, G. (2019). *The Software Heritage acquisition process* (CI-2019/WS/8). UNESCO, Università di Pisa, Inria. <https://unesdoc.unesco.org/ark:/48223/pf0000371017>

CASRAI. (2015). *The CRediT Taxonomy*. <https://casrai.org/credit/>

Chawla, D. S. (2016). The unsung heroes of scientific software. *Nature*, 529(7584), 115–116. <https://doi.org/10.1038/529115a>

Childers, B. R., Fursin, G., Krishnamurthi, S., & Zeller, A. (2016). Artifact Evaluation for Publications (Dagstuhl Perspectives Workshop 15452). *Dagstuhl Reports*, 5(11), 29–35. <https://doi.org/10.4230/DagRep.5.11.29>

Clément-Fontaine, M., Di Cosmo, R., Guerry, B., Moreau, P., & Pellegrini, F. (2019). *Encouraging a wider usage of software derived from research* [Research report]. Committee for Open Science's Free Software and Open Source Project Group. <https://hal.archives-ouvertes.fr/hal-02545142>

COAR, & SPARC. (2019, January). *Good practice principles for scholarly communication services*. <https://www.coar-repositories.org/news-updates/good-practice-principles-for-scholarly-communication-services-2/>

Collberg, C. S., & Proebsting, T. A. (2016). Repeatability in computer systems research. *Commun. ACM*, 59(3), 62–69. <https://doi.org/10.1145/2812803>

Colom, M., Kerautret, B., & Krähenbühl, A. (2019). An Overview of Platforms for Reproducible Research and Augmented Publications. In Bertrand Kerautret, M. Colom, D.

Lopresti, P. Monasse, & H. Talbot (Eds.), *Reproducible Research in Pattern Recognition* (Vol. 11455, pp. 25–39). Springer International Publishing. https://doi.org/10.1007/978-3-030-23987-9_2

Colom, M., Kerautret, B., Limare, N., Monasse, P., & Morel, J.-M. (2015). IPOL: A new journal for fully reproducible research; analysis of four years development. *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, 1–5. <https://doi.org/10.1109/NTMS.2015.7266500>

Dacos, M., Di Cosmo, R., & Pellegrini, F. (2018). *About the proposal for software indicators in Open Science Monitor*. Free and Open Source software working group, National Committee for Open Science. <https://www.ouvri-lascience.fr/wp-content/uploads/2018/11/2018.09.19-About-the-proposal-for-software-indicators-in-OSM.pdf>

DataCite Metadata Working Group. (2017). *DataCite Metadata Schema Documentation for the Publication and Citation of Research Data v4.1* [Application/pdf]. 72 pages. <https://doi.org/10.5438/0014>

Di Cosmo, R. (2020a, May). *Bilatex-Software*. CTAN: /Tex-Archive/Macros/Latex/Contrib/Biblatex-Contrib/Biblatex-Software. <https://www.ctan.org/tex-archive/macros/latex/contrib/biblatex-contrib/biblatex-software>

Di Cosmo, R. (2020b). Archiving and referencing source code with Software Heritage. *ICMS, 12097*, 362–373. https://doi.org/10.1007/978-3-030-52200-1_36

Di Cosmo, R., & Danelutto, M. (2020). [Rp] Reproducing and replicating the OCamlP3I experiment. In *ReScience C* (Vol. 6, Issue 1, p. #2). <https://doi.org/10.5281/zenodo.3936402>

Di Cosmo, R., Gruenpeter, M., Marmol, B., Monteil, A., Romary, L., & Sadowska, J. (2020). Curated archiving of research software artifacts: Lessons learned from the french open archive (HAL). *International Journal of Digital Curation, 15*(1), 16. <https://doi.org/10.2218/ijdc.v15i1.698>

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

Di Cosmo, R., Gruenpeter, M., & Zacchiroli, S. (2018). *Identifiers for Digital Objects: The Case of Software Source Code Preservation*. 1–9. <https://doi.org/10.17605/OSF.IO/KDE56>

Di Cosmo, R., & Zacchiroli, S. (2017, September). Software Heritage: Why and How to Preserve Software Source Code. *Proceedings of the 14th International Conference on Digital Preservation, IPRES 2017, Kyoto, Japan*. <https://hal.archives-ouvertes.fr/hal-01590958>

DINUM. (n.d.). *Browse french public sector source code*. <https://code.etalab.gouv.fr/en/repos>

Eghbal, N. (2016). *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure*. <https://www.fordfoundation.org/work/learning/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/>

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong (YIMING), M., Shou (寿林钧), L., Qin, B., Liu, T., Jiang (姜大昕), D., & Zhou, M. (2020, September). CodeBERT: A pre-trained model for programming and natural languages. *Findings of EMNLP 2020*. <https://www.microsoft.com/en-us/research/publication/codebert-a-pre-trained-model-for-programming-and-natural-languages/>

Fenner, M. (2020). *DataCite Commons—Exploiting the Power of PIDs and the PID Graph*. <https://doi.org/10.5438/F4DF-4817>

Fenner, M., Katz, D. S., Nielsen, L. H., & Smith, A. (2018, May 17). *DOI Registrations for Software*. <https://doi.org/10.5438/1NMY-9902>

French National Committee for Open Science. (2019). *Exemplarity Criteria for funding from the National Open Science Fund through platforms, infrastructures and editorial content*. <https://www.ouvrirlascience.fr/exemplarity-criteria-for-funding-from-the-national-open-science-fund/>

Gil, Y., H. David, C., Demir, I., Essawy, B., Fulweiler, W., Goodall, J., Karlstrom, L., Lee, H., Mills, H., Oh, J.-H., Pierce, S., Pope, A., Tzeng, M., Villamizar, S., & Yu, X. (2016). Towards the Geoscience Paper of the Future: Best Practices for Documenting and Sharing Research from Data to Software to Provenance: Geoscience Paper of the Future. *Earth and Space Science*, 3. <https://doi.org/10.1002/2015EA000136>

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

Gruenpeter, M., & Sadowska, J. (2018). *Moderation of a Software Deposit* [Technical report]. Inria ; CCSD ; Software Heritage. <https://hal.inria.fr/hal-01876705>

Hinsen, K. (2019). Dealing with software collapse. *Computing in Science Engineering*, 21(3), 104–108.

Hinsen, Konrad. (2013). Software Development for Reproducible Research. *Computing in Science and Engineering*, 15(4), 60–63. <https://doi.org/10.1109/MCSE.2013.91>

Hinsen, Konrad. (2020). Computational reproducibility. In *Computation in science*. IOP Publishing. <https://doi.org/10.1088/978-0-7503-3287-3ch6>

Howison, J., & Bullard, J. (2015). Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. *Journal of the Association for Information Science and Technology*, 67(9), 2137–2155. <https://doi.org/10.1002/asi.23538>

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>

Jones, M. B., Boettiger, C., Mayes, A. C., Arfon Smith, Slaughter, P., Niemeyer, K., Gil, Y., Fenner, M., Nowak, K., Hahnel, M., Coy, L., Allen, A., Crosas, M., Sands, A., Hong, N. C., Cruse, P., Katz, D., & Goble, C. (2016). *CodeMeta: An exchange schema for software metadata*. *KNB Data Repository* [Application/ld+json]. KNB Data Repository. <https://doi.org/10.5063/SCHEMA/CODEMETA-1.0>

Katz, D. S., & Clark, T. (2019). *Comparing and analyzing the implementation of data citation and software citation*. <https://doi.org/10.5281/ZENODO.3497122>

Katz, D. S., Hong, N. P. C., Clark, T., Muench, A., Stall, S., Bouquin, D., Cannon, M., Edmunds, S., Faez, T., Feeney, P., Fenner, M., Friedman, M., Grenier, G., Harrison, M., Heber, J., Leary, A., MacCallum, C., Murray, H., Pastrana, E., ... Yeston, J. (2020). The importance of software citation. *F1000Research*, 9, 1257. <https://doi.org/10.12688/f1000research.26932.1>

Katz, D. S., Niemeyer, K. E., Smith, A. M., Anderson, W. L., Boettiger, C., Hinsén, K., Hooft, R., Hucka, M., Lee, A., Löffler, F., Pollard, T., & Rios, F. (2016). *Software vs. Data in the context of citation*. <https://doi.org/10.7287/peerj.preprints.2630v1>

Knoth, P. (n.d.). *CORE*. <https://core.ac.uk/>

Krishnamurthi, S. (2011). *Artifact evaluation for software conferences*. <https://www.artifact-eval.org/>

Krishnamurthi, S., & Vitek, J. (2015). The Real Software Crisis: Repeatability As a Core Value. *Commun. ACM*, 58(3), 34–36. <https://doi.org/10.1145/2658987>

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

Liang, W., & Kai Yong, Z. (2013). Translate gene sequence into gene ontology terms based on statistical machine translation. *F1000Research*, 2, 231. <https://doi.org/10.12688/f1000research.2-231.v1>

Liang, W., & KaiYong, Z. (2013). *Protein Sequence To Gene Ontology Translation System*. Zenodo. <https://doi.org/10.5281/ZENODO.7506>

Making Your Code Citable · GitHub Guides. (n.d.). Retrieved October 7, 2020, from <https://guides.github.com/activities/citable-code/>

Manghi, P., & Bardi, A. (2019). *The OpenAIRE Research Graph—Opportunities and challenges for science*. <https://doi.org/10.5281/ZENODO.2600275>

McDougal, R. A., Morse, T. M., Carnevale, T., Marengo, L., Wang, R., Migliore, M., Miller, P. L., Shepherd, G. M., & Hines, M. L. (2016). Twenty years of ModelDB and beyond: Building essential modeling tools for the future of neuroscience. *Journal of Computational Neuroscience*, 42(1), 1–10. <https://doi.org/10.1007/s10827-016-0623-7>

Meeting, E. G. (2019). *Paris Call: Software Source Code as Heritage for Sustainable Development*. <https://unesdoc.unesco.org/ark:/48223/pf0000366715>

Moher, D., Bouter, L., Kleinert, S., Glasziou, P., Sham, M. H., Barbour, V., Coriat, A.-M., Foeger, N., & Dirnagl, U. (2020). The hong kong principles for assessing researchers: Fostering research integrity. *PLOS Biology*, 18(7), e3000737. <https://doi.org/10.1371/journal.pbio.3000737>

Muench, A., Accomazzi, A., & Holm Nielsen, L. (2017). *Asclepias: Enabling Software Citation & Discovery Workflows*. <https://doi.org/10.5281/ZENODO.803473>

- Pan, X., Yan, E., Cui, M., & Hua, W. (2019). How important is software to library and information science research? A content analysis of full-text publications. *Journal of Informetrics*, 13(1), 397–406. <https://doi.org/10.1016/j.joi.2019.02.002>
- Paskin, N. (2010). Digital object identifier (DOI) system. *Encyclopedia of Library and Information Sciences*, 3, 1586–1592.
- Peroni, S., & Shotton, D. (2020). OpenCitations, an infrastructure organization for open scholarship. *Quantitative Science Studies*, 1(1), 428–444. https://doi.org/10.1162/qss_a_00023
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1986). *Numerical recipes: The art of scientific computing*. Cambridge University Press.
- Ritz, R., Kotaleski, J. H., Strandberg, P., Larsson, A., Lillberg, Y., Chatzopoulou, E., Holm, P., Naeslund, M., Wang, H., & Bjaalie, J. G. (2008). A new software center for the neuroinformatics community. *BMC Neuroscience*, 9(Suppl 1), P89. <https://doi.org/10.1186/1471-2202-9-s1-p89>
- Roberts, K. V. (1969). The publication of scientific fortran programs. *Computer Physics Communications*, 1(1), 1–9. [https://doi.org/10.1016/0010-4655\(69\)90011-3](https://doi.org/10.1016/0010-4655(69)90011-3)
- Rosenthal, D., Baxter, R., & Field, L. (2014). *Towards a shared vision of sustainability for research and e-infrastructures*. <https://www.eudat.eu/news/towards-shared-vision-sustainability-research-and-e-infrastructures>
- Rougier, N. P., Hinsén, K., Alexandre, F., Arildsen, T., Barba, L. A., Benureau, F. C. Y., Brown, C. T., de Buyl, P., Caglayan, O., Davison, A. P., Delsuc, M.-A., Detorakis, G., Diem, A. K., Drix, D., Enel, P., Girard, B., Guest, O., Hall, M. G., Henriques, R. N., ... Zito, T. (2017). Sustainable computational science: The ReScience initiative. *PeerJ Computer Science*, 3, e142. <https://doi.org/10.7717/peerj-cs.142>
- Rousseau, G., Di Cosmo, R., & Stefano Zacchiroli. (2020). Software provenance tracking at the scale of public source code. *Empirical Software Engineering*, 1–30. <https://doi.org/10.1007/s10664-020-09828-5>
- San Francisco Declaration on Research Assessment (DORA)*. (2013). <https://sfedora.org/>
- Seinstra, F., Wallom, D., & Keahey, K. (2015). Editorial. *SoftwareX*, 1–2, 1–2. <https://doi.org/10.1016/j.softx.2015.08.001>
- Shustek, L. J. (2006). What Should We Collect to Preserve the History of Software? *IEEE Annals of the History of Computing*, 28(4), 110–112. <https://doi.org/10.1109/MAHC.2006.78>
- Smith, A. M., Katz, D. S., Niemeyer, K. E., & FORCE11 Software Citation Working Group. (2016). Software citation principles. *PeerJ Computer Science*, 2, e86. <https://doi.org/10.7717/peerj-cs.86>
- Smith, A. M., Niemeyer, K. E., Katz, D. S., Barba, L. A., Githinji, G., Gymrek, M., Huff, K. D., Madan, C. R., Mayes, A. C., Moerman, K. M., Prins, P., Ram, K., Rokem, A., Teal, T. K., Guimera, R. V., & Vanderplas, J. T. (2018). Journal of Open Source Software (JOSS): Design and first-year review. *PeerJ Computer Science*, 4, e147. <https://doi.org/10.7717/peerj-cs.147>

Spagnuolo, M., & Veltkamp, R. (2013). Special issue on executable papers for 3D object retrieval. *Computers & Graphics*, 37(5), A7–A8. <https://doi.org/10.1016/j.cag.2013.04.006>

Spinellis, D. (2003). The decay and failures of web references. *Communications of the ACM*, 46(1), 71–77. <https://doi.org/10.1145/602421.602422>

Stodden, V., LeVeque, R. J., & Mitchell, I. (2012). Reproducible Research for Scientific Computing: Tools and Strategies for Changing the Culture. *Computing in Science and Engineering*, 14(4), 13–17. <https://doi.org/10.1109/MCSE.2012.38>

Stojnic, R., Taylor, R., Karadas, M., Kerkez, V., & Viaud, L. (2019). *Papers With Code*. <https://paperswithcode.com/about>

Sun, S., Lannom, L., & Boesch, B. (2003). *Handle System Overview* (RFC No. 3650). RFC Editor. <https://tools.ietf.org/html/rfc3650>

The Plume Team. (2013). *The Plume project*. https://projet-plume.org/types-de-fiches#logiciel_valide

UNESCO, D. G. (2020). *Preliminary report on the first draft of the Recommendation on Open Science* (CL/4333). UNESCO. <https://unesdoc.unesco.org/ark:/48223/pf0000374409>

UNESCO Expert group meeting. (2019). *Paris Call: Software Source Code as Heritage for Sustainable Development*. <https://unesdoc.unesco.org/ark:/48223/pf0000366715>

Van Noorden, R., Maher, B., & Nuzzo, R. (2014). The top 100 papers. *Nature*, 550–553. <https://doi.org/10.1038/514550a>

Wagner, M. (2017). *Hitting the Bull'S eye with darts: Artifact evaluation in computer science*. <https://doi.org/10.5281/ZENODO.583007>

Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., & others. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3. <https://doi.org/10.1038/sdata.2016.18>

Yates, T. (2019). Source-code access for the long haul. *LWN.Net*. <https://lwn.net/Articles/781246/>

Zhao, R., & Wei, M. (2017). Impact evaluation of open source software: An Altmetrics perspective. *Scientometrics*, 110(2), 1017–1033. <https://doi.org/10.1007/s11192-016-2204-y>

6.3 Task Force Participants

6.3.1 Roberto Di Cosmo (Chair TF SIRS)

Organization(s)

Software Heritage is an open non-profit initiative, launched by Inria in partnership with Unesco, and supported by a variety of stakeholders, including major IT players, government bodies and academic entities. Its stated goal is to collect, preserve and share the source code of all software ever written, with its full development history, building a universal source code software knowledge base. Software Heritage addresses a variety of needs: preserving our scientific and technological knowledge, enabling better software

development and reuse for society and industry, fostering better science, and building an essential infrastructure for large scale, reproducible software studies. With over 9 billion unique source files from over 140 million repositories, it is the largest archive of source code ever built. More info at: <https://www.softwareheritage.org>.

Biography

An alumnus of the Scuola Normale Superiore di Pisa, Roberto Di Cosmo was associate professor at Ecole Normale Supérieure in Paris, then full professor of Computer Science at University of Paris and is currently on leave at Inria, publishing over 20 international journals articles and 50 international conference articles in theoretical computing, functional and parallel programming, and software engineering.

After creating the Free Software thematic group of Systematic, and IRILL, a research structure dedicated to Open Source Software, he got support from Inria to create Software Heritage, with the mission to build the universal archive of all the source code publicly available, in partnership with UNESCO.

Role in Software Heritage: Founder and Director

More info at: <https://www.dicosmo.org/bio.html>

Interest in the SIRS TF

As an old time open access and open source advocate, Roberto strongly believes that research should be open, transparent and reproducible. He has felt the lack of common infrastructures to support this vision, and in particular he feels that software has been ignored for too long as a key pillar of research and Open Science, but there is a positive side to it: we may still be in time to advocate a lean, efficient, open, shared, mutualised, collaborative architecture for scholarly infrastructures for (research) software, and avoid the balkanization and dispersion of efforts that has been seen for decades in other areas. Roberto believes that the SIRS TF is a great opportunity to contribute to this goal.

6.3.2 Jose Benito Gonzalez Lopez (Co-Chair TF SIRS)

Organization(s)

Zenodo is a general-purpose repository that enables researchers, scientists, projects and institutions to share, preserve and showcase multidisciplinary research results (data, software, publications, etc). It is founded in the trustworthy CERN data centre, and it is managed, developed and maintained by CERN, although funding comes also from other sources like: EC through OpenAIRE (main partner), SLOAN foundation, and Arcadia. Zenodo hosts more that 1.5 million records in total, around 100,000 software records (including all versions) and around 350 TBs of files.

Biography

Currently, Jose leads the Digital Repositories Section at CERN that is composed of various teams in charge of developing (Open Source Software) and providing services related to Scholarly Repositories and Open Science Infrastructure. These are the main projects:

- CERN Document Server (CDS), CERN institutional repositories:
 - <https://cds.cern.ch>
 - <https://videos.cern.ch>
- Digital Memory (DM) project, which is charge of:
 - Digitalisation

- Long-term preservation
 - See: <http://digital-memory-project.web.cern.ch>
- Invenio Software:
 - <http://inveniosoftware.org>
- Open and Reproducible Research (ORR), which is in charge of:
 - <https://opendata.cern.ch>
 - <https://reana.io>
- Zenodo, which is in charge of:
 - <https://zenodo.org>

Jose is a Software Engineer by education with a lot of experience on open source software development and project management. Previously to his current position, Jose contributed to the development and later management of the Open Source project Indico⁷⁵.

Interest in SIRS TF

Jose is a strong advocate of Open Science and Open Source Software, as one can derive from his bio. He would like to see Europe moving towards a real Open Science factory which will make research more efficient, fair and accessible to everybody regardless of the affiliation they belong to. It is required to succeed in quite a few areas to achieve such a dream, and one of the fundamental pillars is Software. It is time for Software to be seen as a first-class citizen when it comes to research publications and Jose believes this Task Force is a great opportunity to set the bases on how to achieve just that.

6.3.3 Jean-François Abramatic (Chair WG Architecture)

Organization(s)

The European Open Science Cloud (EOSC) Executive Board advises the European Commission and provides recommendations to develop the EOSC.

Biography

Jean-François is Emeritus Senior Scientist at Inria, the French Research Institute in Computer Science and Applied Mathematics. He is a member of the EOSC Executive Board for the 2019-2020 period and is Chair of the EOSC WG Architecture.

Interest in SIRS TF

In order to evolve research towards the Open Science paradigm, it is essential that publications, data and software are findable, accessible, and reusable. While special attention has been devoted to publications (Open Access), data (FAIR guidelines & Open Data), the efforts towards making research software available to scientists are still in their infancy. It is therefore important to assess the current status of these efforts and plan for deploying initiatives that will give research software its first-class position next to publications and data.

⁷⁵ <https://getindico.io/>

6.3.4 Kay Graf

Organization(s)

ESCAPE⁷⁶ (European Science Cluster of Astronomy & Particle physics ESFRI research infrastructures)) is a EOSC cluster project. Its mission is to establish a single collaborative cluster of next generation European Strategy Forum on Research Infrastructures (ESFRI) facilities in the area of astronomy- and accelerator-based particle physics in order to implement a functional link between the concerned ESFRI projects and European Open Science Cloud (EOSC).

Biography

Kay is senior researcher at the Erlangen Centre for Astroparticle Physics (ECAP) at the University of Erlangen⁷⁷, Germany (FAU) and the general manager there. His research is in the field of astroparticle physics spanning the high-energy physics and astrophysics communities. In addition, he has a long history in software development, coordination and maintenance as the computing and software coordinator for the KM3NeT⁷⁸ neutrino experiment, an ESFRI.

Kay is a member of EAWG, primarily in regards to his work within the EOSC cluster project ESCAPE - where he coordinates the work package on an open science software and service repository (OSSR). OSSR will be a sustainable open-access repository to share scientific software and services to the science community and enable open science. It will house astro-particle-physics-related scientific software and services for data processing and analysis, as well as test data sets, user-support documentation, tutorials, presentations and training activities.

Interest in SIRS TF

Coming from a community where - mostly community-specific - software and data naturally go hand in hand to form the basis of all science products, the handling of complete software lifecycles, the sharing of best practices and the cross-fertilisation via co-developments and re-use of software algorithms and software platforms is his main interest. All those topics are part of the SIRS taskforce - so Kay was eager to work together forming software strategies as one of the pillars of the EOSC.

6.3.5 Miguel Colom

Organization(s)

Miguel Colom represents Image Processing On Line (IPOL), a journal that was founded in 2009 after an ERC advanced grant was obtained by Prof. Jean-Michel Morel at École Normale Supérieure de Cachan (now ENS Paris-saclay). IPOL publishes peer-reviewed articles on signal (mainly image and video) processing, with a special focus on complete mathematical descriptions of the algorithms. The number of publications per year of IPOL is modest, with about 25 papers per year. It's indexed by SCOPUS, DOAJ, and others. An official Impact Factor hasn't been yet obtained, but it's in the Thomson Reuters Emerging Citation Index (a preliminary step before the Impact Factor). Each publication in IPOL includes not only the PDF of the article, but also the source code, both under a free licence.

⁷⁶ <https://projectescape.eu/>

⁷⁷ <https://protect-eu.mimecast.com/s/DnK5CLPWWIRomNTqGFmY?domain=ecap.nat.fau.de>

⁷⁸ <https://protect-eu.mimecast.com/s/mC8jCM9WWhqoRxIJ9XQR?domain=km3net.org>

The large majority of articles have also an online demo where the users can test the algorithms with their own data.

IPOL website: <https://www.ipol.im>

Biography

Miguel is a researcher in image processing at Centre Borelli (ENS Paris-Saclay, France), supervising three PhD candidates on different subjects: detection of falsifications in images based on JPEG artifacts, detection of falsifications based on noise analysis, and design of a new concept of satellite based on irregular interferometric sampling. Before, he worked in noise estimation and denoising of natural and hyperspectral images. His academic background is in applied math and computer science. <http://mcolom.info>

At IPOL, Miguel is a section editor and the designer of the current demo system, a distributed architecture of microservices. He manages the team of engineers that develops and maintains it.

Interest in SIRS TF

Since the beginning of IPOL Miguel has had a strong interest in the quality and long-term durability of the system. This has led to re-implementing several parts until they were considered fully correct. He is really interested in knowing about other platforms like the ones participating in this TF, and to discover what others consider as good practices and great pitfalls.

Also, to know about solutions to common problems that many of the platforms in this task force share: how to archive efficiently, how to ensure reproducibility, how to manage different kinds of granularity when archiving, etc. IPOL has found solutions that IPOL believes are good, but Miguel is interested in knowing different solutions to the same problems by different platforms.

And finally, he thinks this initiative is absolutely needed to gather information on working platforms to arrive at conclusions about good practices that can be useful to others in terms of recommendations. Some kind of "design patterns", but understood as good practices at platform level and for particular tasks (execution of algorithms, archiving, referencing, etc.).

6.3.6 Paolo Manghi

Organization(s)

OpenAIRE is a non-profit legal entity offering networking services and technical services to favour the implementation and adoption of Open Science practices in Europe and beyond. One of the core technical services we offer is the OpenAIRE Research Graph⁷⁹ an open, transparent, metadata collection bringing in all scholarly communication sources worldwide. We collect metadata from around 12,000 sources (Crossref, DataCite, Unpaywall, MAG, ORCID, GRID/ROR, preprints, institutional repositories from OpenDOAR, etc.), organise scientific results in publications, datasets, and software, and interlink them with funders, projects, organizations (and the data sources from which we collect them). The Graph⁸⁰ counts 110 Pubs, 7 Mi datasets, 200K software, 30 funders, 3.5 million

⁷⁹ <https://protect-eu.mimecast.com/s/0nt0CG9WWh1jN1tKCSuT?domain=openaire.eu>

⁸⁰ <https://protect-eu.mimecast.com/s/nWXnCPWWIq0WqfGV4Ca?domain=beta.explore.openaire.eu/>

projects, and around 1Bi relationships between such objects. OpenAIRE is one of the pillars of the European Open Science Cloud⁸¹.

Biography

Paolo Manghi is a (PhD) Researcher in computer science at Istituto di Scienza e Tecnologie dell'Informazione (ISTI) of Consiglio Nazionale delle Ricerche (CNR), in Pisa, Italy. His research areas of interest are today data e-infrastructures for science and scholarly communication infrastructures, with a focus on technologies supporting open science publishing within and across different disciplines, i.e. computational reproducibility and transparent evaluation of science. He is the CTO of the OpenAIRE infrastructure, involved in coordination and research in the H2020 projects OpenAIRE-Connect, OpenAIRE-Advance, OpenAIRE2020. Paolo is also involved in the research infrastructure projects SoBigDataPlus⁸², PARTHENOS, AriadnePlus, RISIS2 and in the European Open Science Cloud projects EOSCpilot, eInfraCentral, EOSC Secretariat, EOSC-Enhance. He is an active member of Research Data Alliance WGs, member of EC projects advisory boards, of the ResearchObject.org⁸³, GreyNet, RD-Switchboard initiative, Open Science Monitor WG for the European Commission, EOSC Architecture WG, GO FAIR GO Inter WG, and World Data System ITO Technical Advisory Committee.

Interest in SIRS TF

Paolo's main research interests are on solutions to Open Science publishing workflows, in order to enable sharing, tracking, monitoring, reproducing, evaluating, rewarding the full scientific process. Recent history on this domain has been tackling these issues starting from Open Access to publications, moving to Open Access to data, FAIR Data, and now for the first time glancing at software as a first class citizen of scholarly communication. He is convinced that this step is necessary and key to move towards an overarching view of science, which is far from being implemented today. When looking at the Open Science roadmap, software should not be intended to be the only missing piece of the puzzle, as services, workflows, facilities are as important as overlooked by scholarly communication, but rather the so far ignored "elephant in the room".

6.3.7 *Melissa Harrison*

Organization(s)

eLife is a non-profit organisation created by funders and led by researchers. Its mission is to accelerate discovery by operating a platform for research communication that encourages and recognises the most responsible behaviours.

eLife works across three major areas:

- Publishing – eLife aims to publish work of the highest standards and importance in all areas of biology and medicine, while exploring creative new ways to improve how research is assessed and published.
- Technology – eLife invests in open-source technology innovation to modernise the infrastructure for science publishing and improve online tools for sharing, using and interacting with new results.

81 <https://protect-eu.mimecast.com/s/K0urCK699T23y2t3zzer?domain=eosc-portal.eu/>

82 <https://protect-eu.mimecast.com/s/IG6aCLPWWIREDRTmZv3-?domain=sobigdata.eu/>

83 <https://protect-eu.mimecast.com/s/q8-jCM9WWhqEyqfQ--o5?domain=researchobject.org/>

- Research culture – eLife is committed to working with the worldwide research community to promote responsible behaviours in research.

eLife receives financial support and strategic guidance from the Howard Hughes Medical Institute, the Knut and Alice Wallenberg Foundation, the Max Planck Society and Wellcome. eLife Sciences Publications Ltd is publisher of the open-access eLife journal.

Biography

Melissa Harrison manages the production department, ensuring the production process, managing content from acceptance to publication and downstream deliveries, is efficient and innovative. Having had editorial and production roles in journals and books at various publishing houses she cemented her preference for workflow, process, and XML within the journal production stage. Melissa chairs JATS4R and contributes regularly to the Force11, Metadata2020, Crossref and JATS communities, campaigning for standardization of data modelling to facilitate the flow of information between users in order to maximize the dissemination and reuse of knowledge. She promotes the machine readability of outputs of open science, including the use of PIDs to link people, institutions, resources, and outputs.

Interest in SIRS TF

As eLife has implemented best practice, open science, and reproducibility standards and led the effort with other publishers, the use and re-use of software as well as giving credit to software "authors" has been very difficult to implement and navigate. Melissa hopes to give the perspective of a publisher trying to do the "right thing" and the issues this involves, and why there is a need to simplify this in order to increase uptake at other journals.

6.3.8 Yannick Barborini

Organization(s)

HAL is the national multidisciplinary open archives platform chosen by French universities, top-ranking universities and research establishments as part of an inter-establishment agreement (2013), to allow their researchers to deposit their scientific production. HAL is operated by Centre for Direct Scientific Communication (CCSD), a joint service unit (UMS 3668) whose supervisory authorities are the CNRS, Inria, INRAE and the University of Lyon, with the financial backing of the Ministry of Higher Education, Research and Innovation (MESRI). HAL is part of the infrastructures included in the "National Plan for Research Infrastructures 2018-2020".

HAL collects and disseminates, via open access, documents produced through research (articles published in peer-reviewed journals, unpublished articles, communications, etc.) pertaining to all scientific fields. To date, HAL contains nearly 730,000 scientific documents and 2,300,000 bibliographic records. Document deposits (94,246 in 2019) are growing by around 20% per year.

The HAL platform also hosts 140 portals of higher education and research institutions. More than one-half of the French universities, research organisations and top-ranking universities. These portals constitute the institutional open archives of these organisations and allow them to implement their Open Access policy and manage their production.

Biography

Yannick Barborini is a software Engineer and has been working at the CNRS since 2005. He has participated in the development of different services offered to the entire scientific community: HAL of course, but also Sciencesconf (conference management platform),

Episciences (overlay journals) and Isidore (search engine in Humanities and Social Sciences). Currently, he leads the team in charge of developing and providing new services to the open archive HAL.

Interest in SIRS TF

Yannick is convinced that research should be open and accessible. Scientific outputs are no longer limited to documents (articles, communications, etc.). HAL has opened software deposits in 2018 thanks to a collaboration between CCSD, HAL Inria and Software Heritage, but they have seen that the management of software items in the archive needs improvement.

Software should be seen as a first-class citizen like other scientific productions and Yannick believes this task force is a great opportunity to share experiences with other partners and to contribute to this goal.

6.3.9 Ville Tenhunen

Organization(s)

Ville Tenhunen is a Finnish member state representative in the EOSC WG Architecture nominated by the ministry of education and culture of Finland. According to Finnish administrative practises this means that he is not representing any official organisation, but represents himself, specifically his own expertise, instead. Ville has worked for the University of Helsinki and now, since the beginning of March this year working for EGI Foundation in Amsterdam as a Data Solutions Architect. He has a temporary contract with the EGI Foundation.

Biography

Ville has worked as a team leader and project manager in the University of Helsinki for more than 12 years. Last major project has dealt with research data and its storages. He has also been active in Finnish national open science and research Initiatives. Additionally, Ville has been co-chair of the Research Data Architectures in Research Institutions IG of the Research Data Alliance (RDA) and is now a member of the Architecture Working Group of the EOSC. Beginning in March 2020 he has worked in the EGI Foundation as Data Solutions Architect. He has also acted as a data manager in the APIKS project where he codes some solutions for an international research project (22 country teams).

In the University of Helsinki and now in the EGI Foundation he has also worked with scientific software in a service provider and service developer roles.

Interest in SIRS TF

Ville is interested in software preservation, discoverability services and PID systems in the context of the research reproducibility and openness. Catalogue services are one point in this manner. Other interests are new forms of the services and software where for example containers, virtual appliances, and functions as a service are discussed.

6.3.10 Michael Wagner

Organization(s)

Schloss Dagstuhl - Leibniz-Zentrum für Informatik (English: Leibniz Centre for Informatics). Schloss Dagstuhl's very general mission is to promote basic and application-oriented research in the field of informatics, to support advanced, scientific vocational training and to further education in the field of informatics, to promote the transfer of

knowledge between research into informatics and application of informatics, and to operate an international forum and research institute for informatics.

Dagstuhl has pursued its mission mainly by facilitating communication and interaction between researchers. Since the very first days of Dagstuhl in 1990, the seminar and workshop meeting program has always been the focus of its programmatic work. In recent years, LZI Schloss Dagstuhl has expanded its operation and also has significant efforts underway in bibliographic services (the dblp computer science bibliography) and in open access publishing.

dblp⁸⁴: The dblp computer science bibliography provides open bibliographic information on major computer science journals and proceedings. Listing more than 5.1 million publications, dblp is the world's most comprehensive open data collection of computer science research articles.

Publishing: Since its beginnings, Dagstuhl has been publishing reports of its seminars and workshops which have been available free of charge, be it on paper or electronically. In reaction to the slow start of the open access idea in computer science and after receiving repeated requests from the community, Dagstuhl started in 2008 an open access publication platform for computer science research. The goal is not so much to become a large publishing house but to establish affordable open access publishing as a viable mode of publication in computer science.

The flagship product of Dagstuhl Publishing is the LIPIcs series⁸⁵, which publishes proceedings of outstanding computer science conferences.

Biography

Michael Wagner has been a member of the scientific staff at Schloss Dagstuhl since 2012. He started there as part of the dblp team, but quickly took on his first task in the growing publishing department at the end of 2012. Since the end of 2017, Michael is now leading the publishing department full time. In addition to the operational publishing business, he is responsible for the development of their software systems together with a colleague.

Interest in SIRS TF

Michael has a strong opinion that science should be open. All contributions - whether text, data, or software - should be freely available to the public. He also thinks that research results should not be in the hands of commercial service providers whose primary intention is to maximize profits. Even if this seems to change slowly in the publishing business (at least the open-access part is slowly increasing; the point affordable publication cost is unfortunately another tough topic), he now sees a great chance not to run into a similar dependency on commercial providers for the publication of data and software but to take an open path with fair and affordable conditions from the beginning. Therefore, Michael is happy to be part of this TF, to learn from the experiences of other members and hopefully to create an open path.

6.3.11 Wolfgang Dalitz

Organization(s)

Zuse Institute Berlin (ZIB): ZIB is an interdisciplinary research institute for applied mathematics and data-intensive high-performance computing. Its research focuses on

⁸⁴ <https://dblp.org/>

⁸⁵ <https://www.dagstuhl.de/dagpub>

modelling, simulation and optimisation with scientific cooperation partners from academia and industry.

Biography

Wolfgang Dalitz is a scientist at Zuse Institut Berlin (ZIB) working in the field of scientific information systems. He has been involved in building mathematical software libraries since the late 1980s. Within the division "Mathematical Algorithmic Intelligence" he leads the Working Group "Open Science and Research Data".

Interest in SIRS TF

Open access to data, codes, methods, and results of scientific research only unfolds its full potential, once it is possible to relate and interconnect them. Based on this, tools can be developed, which are of scientific and social relevance. A prerequisite for this is the existence of suitable scientific infrastructures based on the FAIR principles (Findable, Accessible, Interoperable, Reusable). The partners in this TF support these principles. A common strategy to establish this infrastructure must be the TF's goal.

6.3.12 Jason Maassen

Organization(s)

The Netherlands eScience Centre is the Dutch national centre for the development and application of research software. It is a non-profit organization funded by NWO (the Dutch Research Council) and SURF (the organization for ICT in Dutch education and research). Its main goal is to boost the use of digital methods and research software in Dutch academic research, across all disciplines. To do so, the eScience Centre provides both funding and expertise (in the form of RSEs) to research projects. In addition, it contributes to many topics surrounding research software, such as FAIR software recommendations (fair-software.eu), the research software directory (research-software.nl), software quality guides (the-turing-way.netlify.app), software carpentry courses, etc.

Biography

Jason Maassen is a Technology Lead and involved in many of the projects at the Netherlands eScience Centre that apply parallel and distributed programming to scientific applications. In addition, he guides internal software development at the centre of software sustainability efforts of the eScience Centre, such as the Research Software Directory and fair-software.nl.

Interest in SIRS TF

Jason's main interest is to align the efforts of the Netherlands eScience Centre in the area of FAIR software, software directories, software archiving, software quality guidelines, etc., with those of (potential) European partners and EOSC. So far, the eScience Centre has mostly been active on a national level, with some ad-hoc international cooperation here and there. Jason sees this initiative as an opportunity to widen the scope of these efforts.

6.3.13 Carlos Martinez-Ortiz

Organization(s)

The Netherlands eScience Centre is the Dutch national centre for the development and application of research software. It is a non-profit organization funded by NWO (the Dutch Research Council) and SURF (the organization for ICT in Dutch education and research). Its main goal is to boost the use of digital methods and research software in Dutch

academic research, across all disciplines. To do so, the eScience Centre provides both funding and expertise (in the form of RSEs) to research projects. In addition, it contributes to many topics surrounding research software, such as FAIR software recommendations (fair-software.eu), the research software directory (research-software.nl), software quality guides (the-turing-way.netlify.app), software carpentry courses, etc.

Biography

Carlos Martinez-Ortiz has worked in a wide range of research projects, ranging from digital humanities, life sciences, automatic detection of abnormal energy consumption in buildings, video tracking of dairy cows, modelling high performance storage systems, to segmentation of medical images. In his current role as community manager, he is involved in many of the software sustainability and FAIR software efforts of the eScience Centre.

Interest in SIRS TF

Carlos' main interest is to align the efforts of the Netherlands eScience Centre in the area of FAIR software, software directories, software archiving, software quality guidelines, etc., with those of (potential) European partners and EOSC. So far, the eScience Centre has mostly been active on a national level, with some ad-hoc international cooperation here and there. Carlos sees this initiative as an opportunity to widen the scope of these efforts.

6.3.14 Elisabetta Ronchieri

Organization(s)

The Italian National Institute for Nuclear Physics (INFN), founded in 1951, is a governmental research organization with 20 divisions, spread throughout Italy, 4 national laboratories (Laboratori Nazionali di Frascati, Laboratori Nazionali del Gran Sasso, Laboratori Nazionali di Legnaro and Laboratori Nazionali del Sud), the National Centre for Research and Development in Information Technology (CNAF), based in Bologna, and 2 other national centres (TIPFA, based in Trento, and Lasa, based in Milano). Its mission is to promote, coordinate and fund nuclear, particle and high-energy physics research in Italy. Since its inception, INFN has been developing open ICT innovative solutions for its own advanced needs of distributed computing and software applications. It has a remarkable excellence expertise on Grid and Cloud technologies, having fostered and participated, with leadership roles, to many of the large Projects financed by the EC that eventually led to the realization of the European Grid Infrastructure (EGI). INFN has well established collaborations with the main international Research.

Centres involved in the development of ICT solutions for the scientific world and has been a primary partner of many projects funded by the EC through the FP7 program, in particular EGI_InSPIRE and EGI_Engage. INFN is currently leading the Italian JRU (which involves INAF and INGV) for the participation to EGI_Engage. INFN has been leading one of the three pillars of the EOSC-Hub project, the INDIGO-DataCloud project, under Horizon2020 EU Framework Program for Research and Innovation. The INDIGO-DataCloud has developed a data and computing platform targeting scientific communities, deployable on multiple hardware and provisioned over hybrid (private or public) e-infrastructures.

Biography

Elisabetta Ronchieri is a (PhD) computer science engineer. She has been working at the INFN CNAF since 2001, participating in designing and developing solutions for software maintenance and software quality in various EU projects, such as DataGrid, EGEE, ETICS and EMI. At CNAF Elisabetta is a technologist and member of the software development team, involved in the operations and R&D of the computing infrastructure. She collaborates in the organization of international conferences, such as IEEE NSS/MIC.

Recently, her main research consists of investigating the role of Machine Learning techniques in software engineering issues and in data centre management, collaborating in other EU projects, such as DEEP-Hybrid-DataClouda and IOTwins. Furthermore, she is interested in combining knowledge and data-driven methods for addressing complex problems, like the identification of clinical narrative. Elisabetta collaborates with the secondary schools of Bologna for the technology transfer project and the University of Bologna for data analysis framework.

Interest in SIRS TF

Elisabetta's institute is a supporter of Open Science and Open Source Software. Personally, she is in favour of a reproducible research so that others may verify the findings and build upon them. For this both data and code must be available to researchers and easily executable to obtain the same results. The higher the quality of data and code is, the higher the reliability of the research is. This TF can confer software the proper role inside a research analysis.

6.3.15 Sam Yates

Organization(s)

Swiss National Supercomputing Centre⁸⁶ (CSCS). CSCS manages and provides access to high performance computing systems for Swiss and European researchers and for partner organisations including CERN and MeteoSwiss. In addition to hardware management and user support, CSCS is also engaged in the development of a number of software tools and simulators for HPC environments, and collaborates with many software and infrastructure development projects, both within Europe and around the world.

Biography

Sam Yates is a software engineer at CSCS, primarily engaged in the development of Arbor, a neuron simulation library for HPC systems, as part of the Human Brain Project. His academic background is mainly in pure mathematics, but he has been involved in scientific software development for most of his career, both in industry and academia, with applications in geophysics, telecommunications, and most recently in computational neuroscience.

Interest in SIRS TF

Sam has encountered first hand issues that affect academic projects with a scientific computing component that stem from poor data curation and software engineering. These in turn often arise from mismatches between the systems that recognize and enable researchers, and the practice of software development in a research context. A mature infrastructure for supporting the software component of modern science will, he hopes, help ameliorate these problems, and Sam is very interested in supporting its development.

6.3.16 Moritz Schubotz

Organization(s)

FIZ Karlsruhe – Leibniz Institute for Information Infrastructure researches, develops and operates methods, processes and services for a sustainable information infrastructure. The organization offers data, information and knowledge, software and services via open and legally compliant platforms and makes them searchable, accessible, interoperable and

⁸⁶ <http://cscs.ch/about>

reusable. FIZ Karlsruhe supports the value creation process in science and innovation at all levels and enables research questions to be answered and new ones to be posed. In doing so, the organization follows its guiding principle "Advancing Science". To do this a variety of platforms and services are developed. swMATH⁸⁷, in particular, is a joint effort with Zuse-Institute Berlin. swMATH is a freely accessible, innovative information service for mathematical software. swMATH not only provides access to an extensive database of information on mathematical software, but also includes a systematic linking of software packages with relevant mathematical publications. The intention is to offer a list of all publications that refer to a software recorded in swMATH. In particular, all articles are given, which are included in Zentralblatt MATH (zbMATH). It can be both articles that describe the background and technical details of a program, as well as those publications in which a piece of software is applied or used for research.

Biography

Moritz Schubotz is a senior researcher at the Department for Mathematics at FIZ Karlsruhe - Leibniz-Institute for Information Infrastructure, Germany. As a theoretical physicist and computer scientist he follows his passion for mathematical information retrieval. He applies a bouquet of state-of-the-art computer science technology to academic literature from science, technology, engineering and mathematics. André Greiner-Petter, Philipp Scharpf and Felix Petersen share this special interest and research methods and tools to make mathematical expressions more useful for humans and computers. As a Wikimedia Open Science Mentor Moritz is committed to the FAIR principles and advocates for Open Science in general. Together with the Ph.D. students Cornelius Ihle and Dennis Trautwein he investigates the potential of Blockchain Technology to advance the Open Science movement. He has been an offsite collaborator at NIST (National Institute of Standards and Technology, U.S.A.) since 2014 and was a fellow at the NII (National Institute of Informatics, Japan) from 2017 to 2018. Earlier, he received a Ph.D. in Computer Science from TU Berlin, Germany.

Interest in SIRS TF

Moritz expects that the partners in the SIRS TF are the critical mass to become a nucleus for a fundamental change in research culture, in mathematics and beyond. FAIR software contributes to more effective collaboration in science. The TF describes the foundations for the organization of FAIR software. From there, an infrastructure will evolve so that researchers can focus on their domain-specific problems, and the frameworks to organize archive, reference, describe and credit software artifacts will seamlessly function in the background, like running tap-water.

6.3.17 *Leonardo Candela*

Organization(s)

The National Research Council of Italy (Cnr) is the largest public research institution in Italy, the only one under the Research Ministry performing multidisciplinary activities. Founded as legal person on 18 November 1923, Cnr's mission is to perform research in its own Institutes, to promote innovation and competitiveness of the national industrial system, to promote the internationalization of the national research system, to provide technologies and solutions to emerging public and private needs, to advice Government and other public bodies, and to contribute to the qualification of human resources. In the Cnr's research world, the main resource is the available knowledge which means people, with their skills, commitment and ideas. This capital comprises more than 8.000 employees, of whom more than half are researchers and technologists. Some 4.000 young

⁸⁷ <https://swmath.org/>

researchers are engaged in postgraduate studies and research training at Cnr within the organization's top-priority areas of interest. A significant contribution also comes from research associates: researchers, from Universities or private firms, who take part in Cnr's research activities.

Biography

Leonardo Candela is computer science researcher at the National Research Council of Italy, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo". His research interests are driven by the development of systems and services supporting research infrastructures for science. In particular, he is intertwining virtual research environments, data infrastructures, collaborative working environments, reference models for complex systems, information retrieval, data analytics, data publishing and innovative scholarly communication practices. His research activity is developed by closely connecting research and development. In fact, he has been involved in several EU-funded projects called to develop Digital Libraries & Data Infrastructures and he is the Strategy and Portfolio Manager of the D4Science.org infrastructure.

Interest in SIRS TF

Leonardo is fascinated by open science and he is investing energies and efforts to promote it and implement it. He strongly believes that "publishing" (making public) is the key functionality characterising the open science movement and he is contributing to the development of such a concept. He also believes that systems and solutions for open science must be built according to the "system of systems" paradigm. This leads him to study the approaches for "data publishing" by considering the complementary perspectives and offerings of journals and repositories. Research software represents another important research artifact capturing his "publishing"-related interest. The SIRS task force is a unique opportunity to discuss his understanding on the matter and bring the ideas on research software management according to open science practices forward.

6.3.18 Martin Fenner

Organization(s)

DataCite is a leading global non-profit organization that provides persistent identifiers (DOIs) for research data, research software and other research outputs. Organizations within the research community join DataCite as members to be able to assign DOIs to all their research outputs.

Biography

Martin Fenner has been the DataCite Technical Director since 2015. From 2012 to 2015 he was the technical lead for the PLOS Article-Level Metrics project. Martin has a medical degree from the Free University of Berlin and is a Board-certified medical oncologist. He co-chairs the Force11 Software Citation Implementation WG and the RDA/FORCE11 Software Source Code Identification WG, and is a member of the EOSC Architecture WG.

Interest in SIRS TF

Martin is particularly interested in addressing the needs for persistent identification and standardized metadata for research software, and the interlinking of research software, publications, research data, researchers, research organizations and funding.

6.3.19 *Eric Jeangirard*

Organization(s)

scanR is a service offered by the French Ministry of Higher Education, Research and Innovation. scanR is a tool for exploring the research and innovation landscape in France. It aims to help understand who the actors of research and innovation in France are, to promote their work. scanR is intended for the entire French society in a logic of transparency of work largely supported by public funds. It also aims to promote access for all to the latest research and innovation developments in order to stimulate public debate. Finally, scanR intends to contribute to the intensification of links between different actors (belonging to different fields of research or status), which are an important vector for boosting this activity.

Biography

Eric Jeangirard is a Data Scientist working for the French Ministry of Higher Education, Research and Innovation since 2018. In particular, besides scanR, he has been involved in the design and implementation of the French Open Science Monitor. In the team, Eric uses recent techniques in machine learning and software application deployments to build efficient and inexpensive tools for the world of higher education and research.

Interest in SIRS TF

The development of Open Science to make the results of scientific research open to all, without hindrance, without delay, without payment is one of the commitments of the French Ministry through the National Plan for Open Science. The implementation of the French Open Science Monitor and scanR are part of the action points for its development. The establishment of scholarly infrastructure to enable the identification and management of standard metadata for research software, enabling them to be located in the research ecosystem through links with researchers, entities, publications and funding is a major challenge for research software.

Getting in touch with the EU

IN PERSON

All over the European Union there are hundreds of Europe Direct information centres.

You can find the address of the centre nearest you at: https://europa.eu/european-union/contact_en

ON THE PHONE OR BY EMAIL

Europe Direct is a service that answers your questions about the European Union.

You can contact this service:

- by freephone: 00 800 6 7 8 9 10 11 (certain operators may charge for these calls),
- at the following standard number: +32 22999696, or
- by email via: https://europa.eu/european-union/contact_en

Finding information about the EU

ONLINE

Information about the European Union in all the official languages of the EU is available on the Europa website at: https://europa.eu/european-union/index_en

EU PUBLICATIONS

You can download or order free and priced EU publications from:

<https://op.europa.eu/en/publications>. Multiple copies of free publications may be obtained by contacting Europe Direct or your local information centre (see https://europa.eu/european-union/contact_en)

EU LAW AND RELATED DOCUMENTS

For access to legal information from the EU, including all EU law since 1952 in all the official language versions, go to EUR-Lex at: <http://eur-lex.europa.eu>

OPEN DATA FROM THE EU

The EU Open Data Portal (<http://data.europa.eu/euodp/en>) provides access to datasets from the EU. Data can be downloaded and reused for free, for both commercial and non-commercial purposes.

The Task Force on Scholarly Infrastructures of Research Software, as part of the Architecture WG of the European Open Science Cloud (EOSC) Executive Board, has established a set of recommendations to allow EOSC to include software, next to other research outputs like publications and data, in the realm of its research artifacts. This work is built upon a survey and documentation of a representative panel of current operational infrastructures across Europe, comparing their scopes and approaches.

This report summarises the state of the art, identifies best practices, as well as open problems, and paves the way for federating the different approaches in view of supporting the software pillar of EOSC.

Research and Innovation policy



Publications Office
of the European Union