IJDC | *Conference Pre-print*

Curated Archiving of Research Software Artifacts: Lessons Learned from the French Open Archive (HAL)

Roberto di Cosmo
Inria, Software Heritage,
University of Paris, France

Morane Gruenpeter
Inria, Software Heritage,
University of Paris, France

Bruno Marmol
Inria, Software Heritage,
University of Paris, France

Alain Monteil
IES Inria, France

Laurent Romary
Inria, France

Jozefina Sadowska
IES Inria, France

Abstract

Software has become an indissociable support of technical and scientific knowledge. The preservation of this universal body of knowledge is as essential as preserving research articles and data sets. In the quest to make scientific results reproducible, and pass knowledge to future generations, we must preserve these three main pillars: research articles that describe the results, the data sets used or produced, and the software that embodies the logic of the data transformation.

The collaboration between Software Heritage (SWH), the Center for Direct Scientific Communication (CCSD) and the scientific and technical information services (IES) of The French Institute for Research in Computer Science and Automation (Inria) has resulted in a specified moderation and curation workflow for research software artifacts deposited in the HAL the French global open access repository. The curation workflow was developed to help digital librarians and archivists handle this new and peculiar artifact - software source code. While implementing the workflow, a set of guidelines has emerged from the challenges and the solutions put in place to help all actors involved in the process.

Submitted 12 December 2019 ~ *Accepted* 19 February 2020

Correspondence should be addressed to Morane Gruenpeter, Email: morane@softwareheritage.org

This paper was presented at International Digital Curation Conference IDCC20, Dublin, 17-19 February 2020

The *International Journal of Digital Curation* is an international journal committed to scholarly excellence and dedicated to the advancement of digital curation across a wide range of sectors. The IJDC is published by the University of Edinburgh on behalf of the Digital Curation Centre. ISSN: 1746-8256. URL: http://www.ijdc.net/

Introduction

Modern research relies on software, but it has only gained recognition recently. While strategies for articles and even data preservation are already the norm, software is still a unique artifact for which it is rare to find dedicated deposits and preservation mechanisms in institutional repositories (Milliken, 2019). We need to preserve source code alongside scientific articles and datasets to scaffold future work on top of these open science pillars. As declared on the Inria /UNESCO Paris call:

> 'Recognise software source code as a fundamental research document on a par with scholarly articles and research data;' (UNESCO-Inria, 2019)

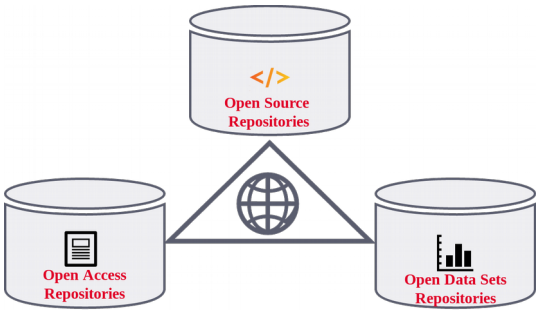

Figure 1. The Open Science pillars for sharing articles, data and software.

Today, software is still too often considered as just data, even though data is gathered through observations or experiments, whereas software is a product of human ingenuity, written by authors and contributors, and embodying the logic of the data transformation. As mentioned in (Alliez et al., 2019), it is challenging to determine who should get credit for the software and which authority has the capability of doing so. Software can be designed and developed by a large number of contributors with a rich development history and a complex web of dependencies. This is why software source code should be considered a research output category of its own. We need to establish preservation strategies to capture both the scientific knowledge it contains and the metadata to comprehend its context.

To ensure preservation of source code, three actors in the French and international research community have collaborated to provide a place for researchers to deposit their source code.

Hyper Articles en Ligne a.k.a HAL

The first actor in this collaboration is HAL, the French national open access repository, created in 2000 by the French National Centre for Scientific Research (CNRS[1]) and maintained by the Center for Direct Scientific Communication (CCSD), [2] destined to provide tools for archiving and dissemination of scientific outputs openly. HAL is a repository where researchers can deposit their academic outputs compliant with their copyrights.[3] Since its creation, HAL has supported different types of deposits: publications, documents (e.g. pre-prints and reports),

[1] http://www.cnrs.fr/en/cnrs
[2] CCSD: a combined service unit (UMS3668) (https://www.ccsd.cnrs.fr/en/)
[3] https://u-paris.fr/hal-archives-ouvertes/ (accessed on 28.11.2019)

academic work (e.g. theses) and research data (e.g. images, videos). HAL's goal is to make research as accessible and open as possible.

IES Inria

Another collaborator in this effort is Inria, the French National Institute for computer science and applied mathematics. Inria, created in 1967, currently hosts in its teams over 3000 researchers[4] and supports the creation of a broad spectrum of open source software, including award winning projects such as Coq, OCaml, and Sckit-Learn.

The research center has a dedicated scientific and technical information service, denoted IES-Inria, which played a major role when specifying the new type of research output: software source code, shown in (Barborini et al., 2018).

Software Heritage (SWH)

The third collaborating initiative is Software Heritage, a nonprofit organization whose goal is building the Library of Alexandria for software source code by collecting, preserving and making the source code available in the long term, as detailed in (Abramatic et al., 2018) and (Di Cosmo and Zacchiroli, 2017).

Software Heritage initiated this collaboration, due in part because of its primary goal and practical knowledge of how to implement software preservation workflows.

These three collaborators designed and implemented a complete workflow dedicated to research source code artifacts that involves three major steps:

1. depositing software source code on HAL's platform
2. moderating and curating the deposit by a certified IES-Inria moderator
3. sharing the deposit and pushing the deposit to the SWH archive

Thanks to this fruitful collaboration software deposits were integrated into the document types supported by HAL, in September 2018.

In this article, we detail the workflow for curating the deposit of software artifacts in the HAL open access repository and the guidelines put in place for the people involved in the process. We describe the transition from test phase to the global integration. Then, we share the lessons learned from the implementation and usage of the source code deposit and the specified workflow. We conclude by presenting the next steps in our software deposit roadmap.

A workflow for curating the deposit of software artifacts

From the earliest open repositories to now, moderation has been a key part of the submission workflow when self-archiving research outputs. One prominent example is ArXiv[5], founded in 1991 and operated by the Cornell University.

Today, there exist platforms that offer source code deposit, such as Zenodo and Figshare, but do not have any pre-submission checks for the self-archived content. HAL chose to follow

[4] https://www.data.gouv.fr/fr/organizations/inria/ accessed on 28.11.2019
[5] https://arxiv.org/ accessed on 28.11.2019

ArXiv's example[6] and implement a sophisticated moderation workflow in order to ensure that quality metadata is attached to every deposit into the platform.

In order to extend the existing HAL moderation workflow to support deposits of research software, a similar workflow had to be implemented to handle the following aspects:

- artifacts attribution
- classification
- compliance with metadata requirements
- and appropriate content

As described in detail in (Alliez et al., 2019), keeping the humans in the loop, similarly to the ArXiv moderation (ArXiv moderators, 2019), is essential to have quality metadata and better credit attribution.

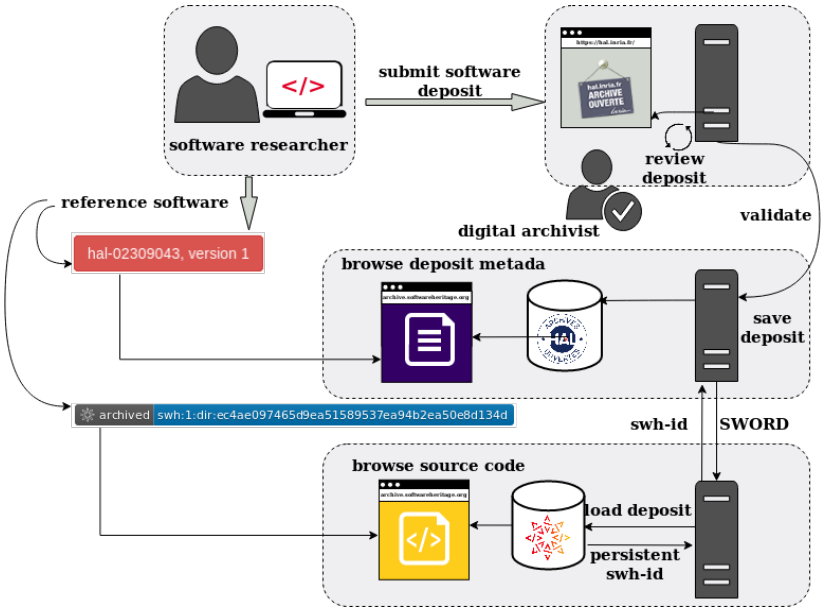

Figure 2. The deposit workflow on the HAL platform and archiving into SWH

The submission form

Contributors must fill out a descriptive metadata form on submission, to ensure the most accurate information about the source code is captured. The metadata is used for moderating the submission and is preserved with the software in both HAL and the SWH archive.

The design of the form was adapted from the pre-existing deposit form for scientific articles, see figure 3 where you can choose the software type and add a software license. The HAL metadata schema included terms that are applied to all deposits (e.g. author, title and keywords, etc.) However, it wasn't sufficient to describe software artifacts.

[6]https://arxiv.org/help/moderation accessed on 28.11.2019

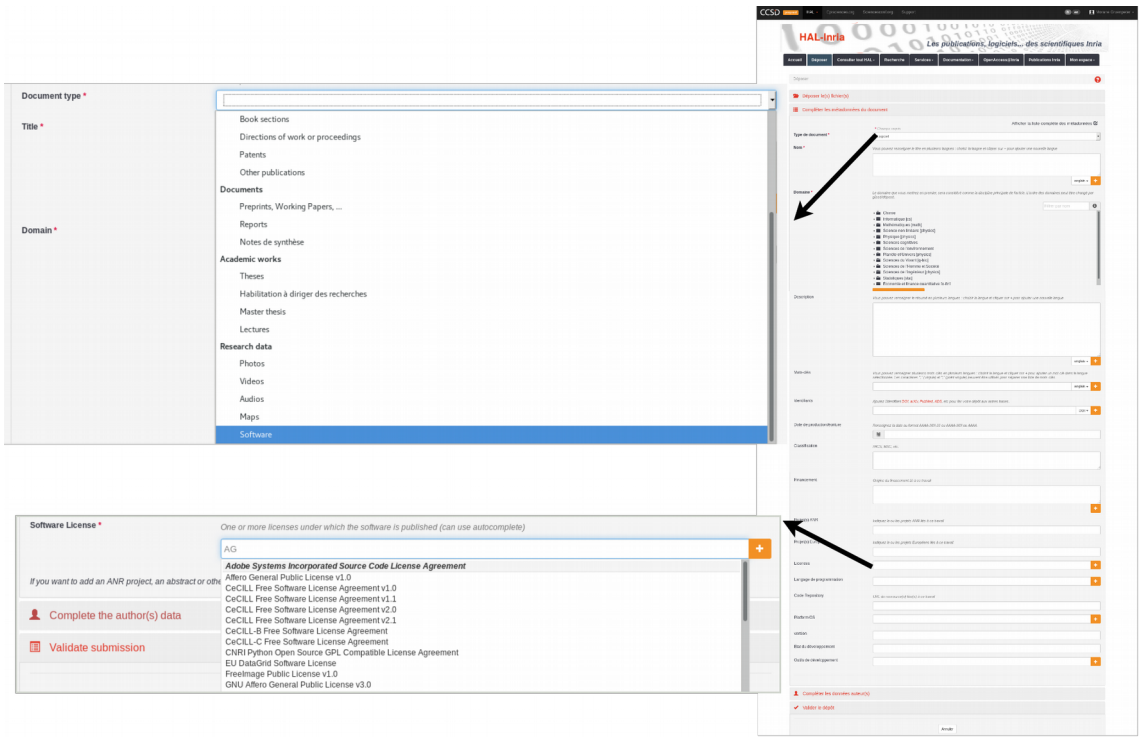

Figure 3. The software deposit form on the HAL-Inria instance platform

Software requires more specific elements in addition to these to adequately describe its complexities. We researched the software vocabulary landscape for a vocabulary adapted to scientific software, and we found that the CodeMeta vocabulary was a perfect fit. A refinement of the schema.org classes SoftwareApplication and SoftwareSourceCode, it provides a convenient bridge with linked data and the semantic web. In addition, the core metadata for software is compliant with existing standards like TEI and Dublin Core.

In Table 1, we compare the HAL metadata terms with the following legend:

- regular text: term that already existed for an article deposit
- **bold text:** term that is mandatory with the software source code deposit
- *italic text*: term specifically added for software

Table 1. The descriptive metadata to ensure an accurate description of the source code artifact

Software source code		
HAL metadata terms	**CodeMeta terms**	**TEI**
HAL ID	identifier	idno:halId
SWH ID	identifier	idno: swhid
Document type		classCode: halTypology
name	name	title

(Table 1 Continued) **Software source code**

HAL metadata terms	**CodeMeta terms**	**TEI**
Domain	applicationCategory	classCode:halDomain
Description	description	note:description
Keywords	keywords	keywords
Identifiers	identifier	idno:doi idno:arXiv ...
Production date	dateCreated	date:whenWritten
[deposit date]	datePublished	date:whenReleased
Classification		classCode: classification
ACM Classification		ClassCode:acm
Comment	releaseNotes	note:commentary
Internal note	referencePublication	localRef:refinterne
Project/Collaboration		
See also	relatedLink	seeAlso
Contract, financing	funding	funder
ANR project(s)	funding	funder:ref="#projAnr"
European project(s)	funding	funder:ref="#projeurop"
softwareLicence	license	availability:licence
programmingLanguage	programmingLanguage	note:programmingLanguage
codeRepository	codeRepository	codeRepository
platform	operatingSystem	note:platform
version	softwareVersion	note:version
developmentStatus	developmentStatus	
runtimePlatform	runtimePlatform	
file		file
author	author	author
Related data	supportingData	

The Software deposit guidelines

We identified that a set of requirements beyond this submission form was needed to curate software deposits. To this end, we have created two user guides, one for the researchers that submit the software (Gruenpeter and Sadowska, 2018a), and one for the digital archivist in charge of the moderation (Gruenpeter and Sadowska, 2018b).

When researchers want to archive and share their code as a citable artifact, they can submit it to either the main HAL instance[7] or on a specific institutional instance (e.g. Inria's instance[8]). No matter where the deposit lives, all materials are discoverable on the central HAL instance.

In the current implementation, researchers must provide a compressed archive, containing the source code (mostly text files).

Researchers are asked to prepare the software source code archive, before submission, by adding the following files:

- AUTHORS
- LICENSE (Preferably from the SPDX referential catalog[9])
- README - Elements that we require and recommend to be included in the README file were taken from the "Best Practices on How to Release Software" from (Raymond E. S., 2000)
    - MUST include:
        - name of the software/project
        - a brief description of the project
    - SHOULD include:
        - project website or documentation pointer
        - authors/credits list (if not in AUTHORS file)
        - license (if not in LICENSE file)
        - Contact and support
    - CAN include:
        - list of features
        - developer's build environment
        - build, installation, requirements - how to run the code
        - usage - how to use the source code
        - recent project news
        - visual

[7] The main HAL instance on hal.archives-ouvertes.fr
[8] The Inria instance on hal.inria.fr
[9] spdx.org: The SPDX License List is a list of commonly found licenses and exceptions used in free and open source and other collaborative software or documentation.

To help researchers and ensure uniformity of the submitted metadata, we have added auto-completion for the license property, using normalised terms directly extracted from the SPDX reference standard, developed and maintained by the software industry.

Curating software - including humans in the loop

The professionals curating deposits into HAL are librarians and archivists. They are employed by specific institutions, if the institution has authority over its institutional repository (e.g. Inria and University of Lorraine) or directly by the CCSD which operates HAL and all attached services. The curation of deposited digital artifacts is one of the roles they assume as information experts. Most of these librarians and archivists have a background in academic institutions, and curating these deposits is one of their key responsibilities.

The process of moderating source code deposits requires human intervention, which leads to direct interactions between the submitting researcher and these curators.

These consultations center around the metadata attached to the deposit rather than the source code itself, although a mild inspection of the code is done to ensure the metadata describing it is correct.

Functional or scientific evaluation of the artifact are not in the scope of the moderation process put in place for HAL software deposit: that role belongs not to repositories or archives, but to reviewing committees. These committees might review software to verify installation instructions, documentation, functionality and tests. Examples of how this is done can be seen looking at the Information Processing On Line Journal (IPOL team, 2019), that has been publishing software implementing image processing algorithm for almost a decade, or the Journal of Open Source Software, which includes many of these criteria in their review guidance documentation (JOSS team, 2019).

A growing number of conferences[10] have an artifact Evaluation Committee (AEC) that evaluates the software artifacts associated to the submitted articles. For example, the POPL conference has an artifact Evaluation Process (AEP) since 2015, where the AEC checks for consistency with the paper, good documentation, and reusability for further research[11]. Artifact evaluation is now also encouraged by the Association of Computing Machinery (ACM) with the ACM badges[12], which can be awarded if the evaluation criteria are met.

By contrast, the HAL moderation process only verifies the accuracy of the descriptive information regarding a deposited software source code artifact and the accuracy of its attribution. During the process, the digital archivist also inspects the artifact to check that the content included in the archive does fit a research deposit. The deposit will not be reviewed in the academic sense of the term, so the functionality of the source code or its reproducibility are not verified.

In figure 4, the contribution and moderation workflow is detailed with the actions that each actor will make to ensure proper archiving of source code. First, the contributor (which can be a researcher, a team member or an institutional representative in charge of the contribution) will prepare the artifact as detailed in the software deposit guidelines, upload the compressed archive, and add metadata on the submission form. Then, the moderator will review the deposit by verifying that the metadata matches the artifact itself and the values in the submission form. The moderator will also check for extraneous content, for example videos, images, or other material that is unlikely to be part of a software source code bundle. If the contributor has listed a code repository, the moderator will verify that the authors of the deposit and in the code repository are the same, even if using pseudonyms, to ensure due credit is given.

[10] See the list maintained at https://www.artifact-eval.org/
[11] https://popl19.sigplan.org/track/POPL-2019-artifact-Evaluation
[12] https://www.acm.org/publications/policies/artifact-review-badging

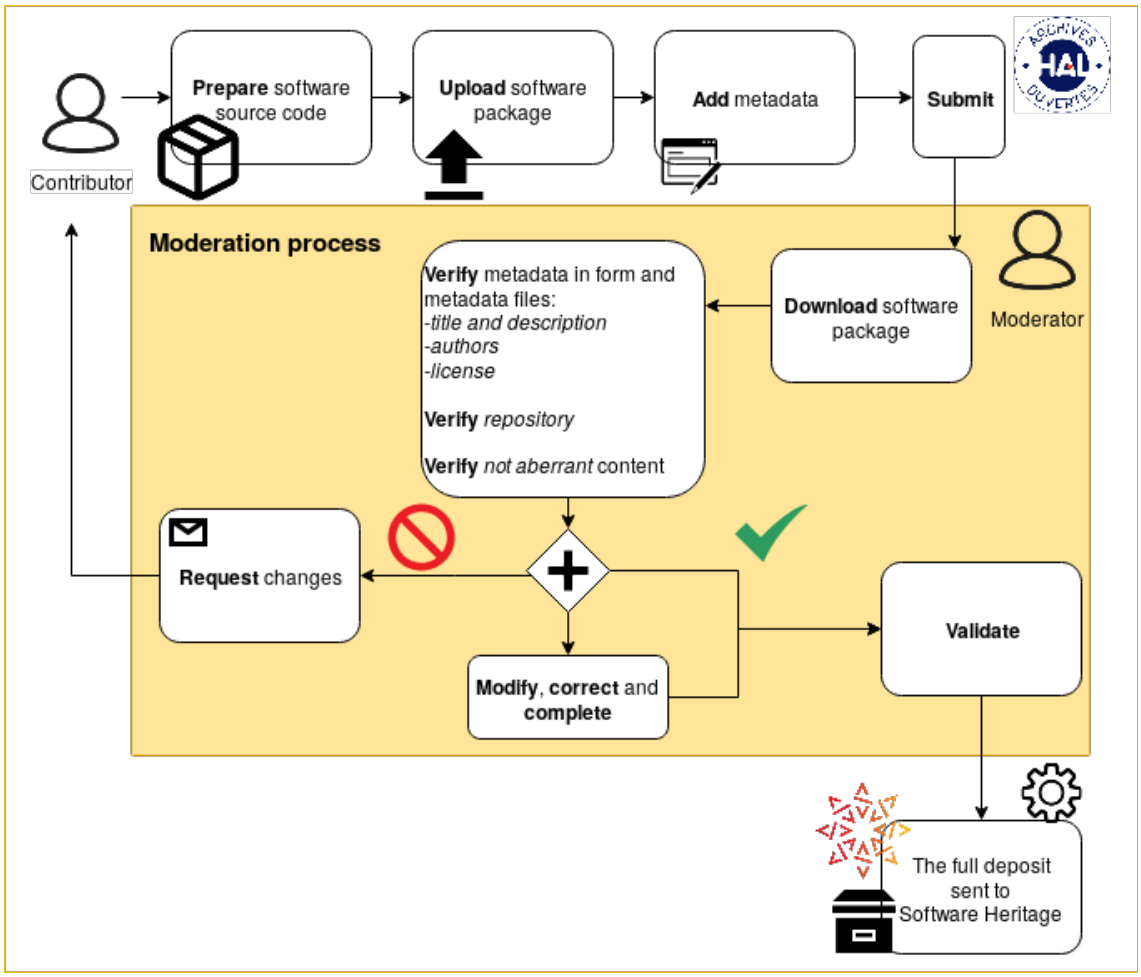

Figure 4. The moderation process when reviewing a software artifact for archival

Our experience over the first two years of operation shows that, with the support of the guidelines, the software moderation process does not add greatly to the workload of digital archivists, and can be performed by digital archivists.

The IES-Inria and CCSD teams, which play the role of digital archivists for HAL platform, are used to working with articles, reports and other textual deposit types. The software deposit was very different from that which they were used to review. When establishing the requirements for a software deposit, we realized that there is no need, at this point, to act as an AEC and verify the functionality, the quality and reproducibility of the artifact itself.

Therefore, the main actions the digital archivist performs while reviewing software deposit are:

- detecting extraneous or abusive content (illegal or harassing),
- verifying consistency between the metadata and the software source code itself,
- completing or correcting the deposit metadata if needed.

During the review process, the digital archivist can request modifications to the deposit from the contributor using a request ticket system, providing a channel with pre-written responses for identified recurrent issues.

Communicating with the contributors and researchers, during the test phase, over their deposits enriched the curation process and helped creating better specifications for the HAL software source code deposit guidelines.

Transferring source code from HAL to SWH

The Hal platform had already implemented transfers of content to Arxiv via the SWORD protocol, available on HAL's documentation (CCSD Development team, 2017). The same integration between HAL and SWH has been designed and implemented using the same protocol.

The deposit is automatically pushed to SWH after a moderator has validated the submission. On reception the deposit is verified by an automated tool. If the verification passes, the deposit is published on HAL's platform and the deposit is scheduled for ingestion in the SWH archive. Otherwise, a detailed error is returned.

The SWORD 2.0 (Jones and Lewis , 2013) implementation provides the technical interface between a client (HAL) and a server (SWH) to push deposits of software source code with associated metadata, available on the API documentation (Software Heritage Development team, 2017).

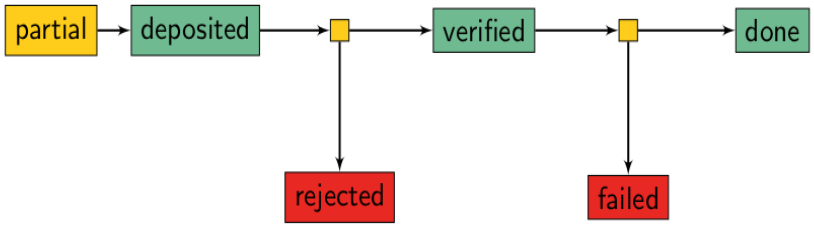

Figure 5. The deposit status on the Software Heritage archive

First, when a deposit arrives to SWH, an automated verification insures the artifact contains a compressed archive and the associated metadata. After it is verified, the ingestion of the content into the archive starts, as illustrated in figure 5.

During the ingestion of the software artifact, SWH computes an intrinsic identifier, the SWH-ID, using a cryptographic signature of the software artifact, see Di Cosmo, Gruenpeter and Zacchiroli (2018) for a detailed explanation.

This SWH-ID does not depend on a resolver and allows to identify the deposit no matter the future developments and organizational changes. This SWH-ID is presented alongside the HAL-ID on the Software artifact view on the HAL platform.

The software view

The deposited software artifacts are accessible on the HAL platform in a specific software view, as presented in figure 6, with the complete metadata record and offers several services:

- TEI, DublinCore or Bibtex exports
- the link to the browsable source code on SWH, in figure 7

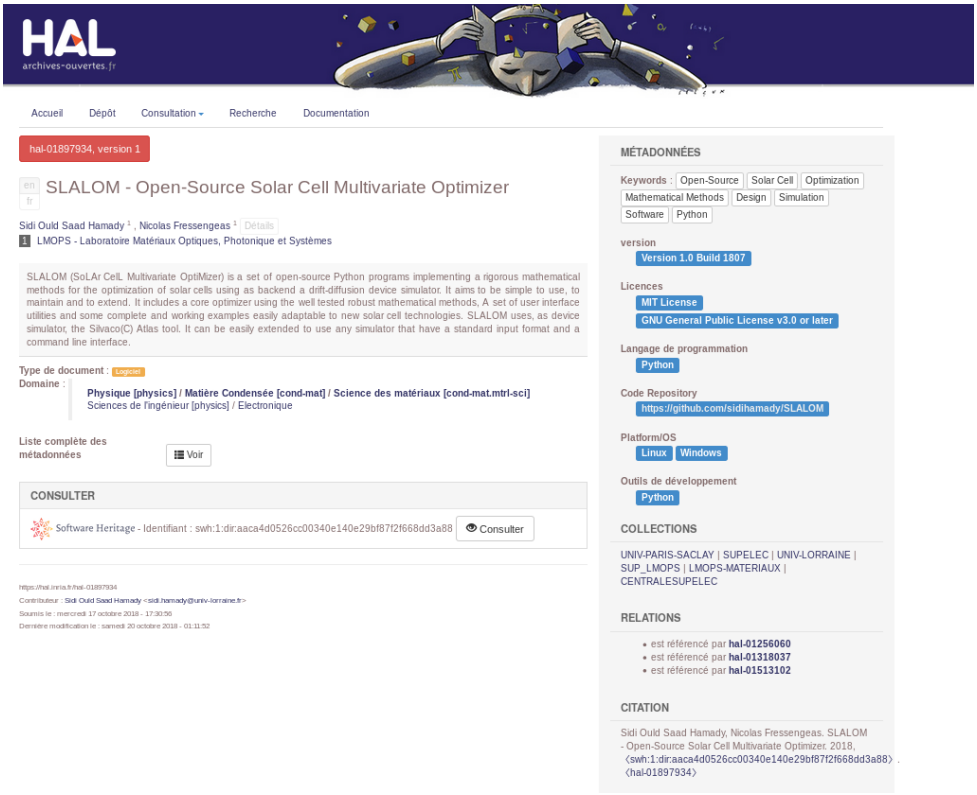

Figure 6. A software deposit on the HAL platform

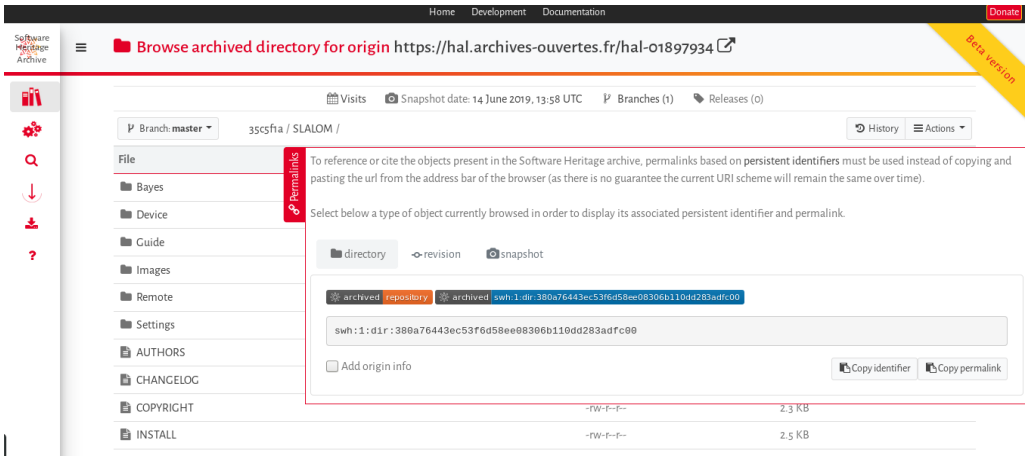

Figure 7. The deposit's browsable source code on the SWH web-app

From test phase to global integration

After we defined the specifications and requirements for the software source code deposit, the CCSD and SWH engineers built a prototype which was only accessible on HAL-Inria, and provided a first test of a software deposit and the HAL to SWH integration.

Between February 2018 and July 2018, a panel of researchers were invited to test the software deposit, described in (Barborini et al. 2018). Their feedback was integrated into the

final version and contributed to improve the deposit guidelines. Throughout this period, the IES-Inria digital archivists tested the moderation process. With their input, a few ergonomic changes were made to the moderation view and the standardised responses to request changes from submitters. During the test phase, 12 software artifacts were uploaded.

The test phase was incredibly valuable for creating and consolidating specific guidelines for the contributors and for the moderators.

The official opening of the software artifact deposit for all HAL instances was on the 25th of September 2018 and was reported by the local press.

In December 2019, we can count 80 source code deposits and 98 software records deposits, which is a promising start for curating software deposits as a research output.

Deposits without source files

During the test phase, researchers could also deposit metadata records about source code without the source code itself, similar to "bibliographic records." Occasionally, users have chosen to deposit only descriptive information about a software artifact, because they needed the reference to the software record in their activity reports. The clear drawback is that it is impossible for the digital archivist to check the information deposited. One approach is to prevent software deposit without the software source code itself, which would be a compressed static archive without its development history.

While this approach is reasonable for researchers that do not use collaborative development platforms, it turns out to be an annoyance for those that have made their software source code available online, or even archived it already in SWH.

The next version of software deposit in HAL should allow to provide just the link to SWH, or to the code repository, where it will be possible for SWH to fetch the source code instead of uploading a compressed archive, lowering the barrier for software deposits into HAL.

Lessons learned

Open issues

We have handled a variety of deposits since the service has been open, and discovered interesting corner cases that led us to evolve our software deposit policy:

- Collective authorship: sometimes we receive the request to use the team name as the software author, instead of providing the full list of contributors. We are evaluating the possibility of a solution of supporting one collective author, and at the same time have a sort of "corresponding author" for managing the deposit; Also, we keep in mind that authorship can be established only with a clear link between a person and a deposit, which is difficult with the collective authorship;
- Legacy software: software that was created a long time ago should be archived in its original state, but it would be useful to add extra information to describe its origin. We are working on a dedicated standard for this particular use case;
- Software collections: sometimes researchers try to deposit a single archive containing many different software tools or software libraries;
- Research experiments that do not really qualify as a software tool on their own; for this particular use case, the researchers usually only need long term archival and intrinsic identifiers: we plan to refer them to the dedicated guidelines for source code archival and reference available on the Software Heritage website (Di Cosmo, 2019);

- Software source code deposited that include large datasets, instead of a reference to a separate data deposit.

The importance of a software license

During the test phase the license of the software wasn't a mandatory metadata and the user form didn't instruct users how to choose a license. As could be expected, this led to deposits with many variations in the software license names and even deposits without a license. Hence we made the license mandatory, and we now provide autocompletion for license names using the standard list developed by the SPDX project of the Linux Foundation for a large consortium of industry players.

Publishing versus sharing

Research software has been around for decades, and some research institutions have a long experience in managing it as a valuable output of research (Alliez et al., 2019), but only very recently attention has started to grow in the broader scholarly ecosystem. This new interest has spawned a rich discussion about what actually could be a software publication. In this context we would like to stress the importance of remembering that in the scholarly world there is a precise semantics attached to the term publication: an academic publication is a research result that has been qualified through some form of peer review; a result that has been simply shared, for example by making it available somewhere on the Internet, is usually not regarded as a publication[13].

When we come to software, that is in its vast majority developed outside of academia, and in particular to open source software, it is common practice to share it broadly on code hosting platforms like GitHub, GitLab, and many other ones, but this act of sharing does not carry the same meaning as the act of academic publishing, and code hosting platforms do not play at all the same role as publishers in the academic world.

Hence we should refrain from using the term "publication" when we talk about software that is simply shared on the Internet, even when its source code is deposited on institutional archives. The research community is still exploring how to exactly handle software when it comes to credit and academic recognition, with various ongoing experimentations like the AEC, IPOL, the Journal of Open Source Software (JOSS team, 2019; Smith et al., 2018), the Dagsthul DARTS series[14], ACM Badges, etc: it is up to researchers to reach an agreement on this very sensitive issue.

For this reason, in the metadata for software deposited via HAL, we do not indicate HAL as a publisher.

Keeping the human in the loop

Even if we do not know yet what should qualify as a software publication, we do know that we need quality metadata to describe research software, and to be used for citing software artifacts. We argue that this requires human intervention, and that it is not enough to just share software on code hosting platforms like GitHub, or self-archive it on repositories like Figshare or Zenodo.

This is why for deposit in HAL and archival in SWH a moderation process is put in place: to ensure that the deposit is a software artifact that reflects a scientific endeavour and that due credit is attributed to all authors of the software without a quality and functionality review of the source code.

[13]In BibTeX, for example, the entry *unpublished* is used for material that has not been formally published.
[14]https://www.dagstuhl.de/publikationen/darts/

Software Identification, reference and citation

We follow the Software Citation Principles (Smith et al. 2016) to create a citation for software deposits into HAL. In figure 8. we have proposed a citation format containing metadata submitted with the software deposit, which is already available on the HAL platform.

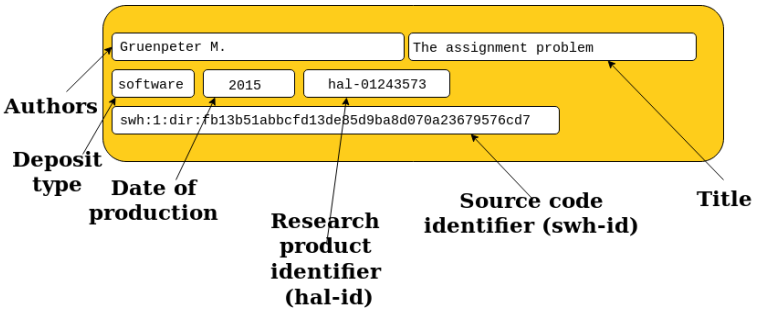

Figure 8. The proposed citation for software artifacts on the HAL platform.

In the citation format, two identifiers are used: the first for the research product, the HAL-ID and the second for the software source code itself with the SWH-ID of the root directory containing the complete development tree. While the HAL-ID identifies the metadata and thus the attribution of the research product, the SWH-ID references the exact version of software source code associated to the deposit. Each identifier caters to different use cases.

At the moment we are working on a proposal for a specific BibTex @software entry as it was already introduced in BibLateX (Kime, Wemheuer and Lehman, 2019) to provide a better BibTex export on the HAL platform. The proposal is developed with Inria's citation working group and will be shared with FORCE11's Software Citation Implementation WG[15] and RDA Software Source Code IG[16] for feedback.

The proposal development is public and can be viewed and commented on its dedicated repository[17].

Conclusion

Decades of experience handling research projects at Inria have shown that a proper moderation process is important to ensure the high quality of the metadata associated to the research software artifacts. To support this process, the collaboration between Software Heritage, Inria and HAL has created tools and guidelines that enable digital archivists to efficiently handle research software deposits, and offers to the HAL users dedicated services for helping preserving and disseminating their software artifacts. We believe that this is an important step forward in the long journey to make software a first class research output in the scholarly ecosystem. On the HAL-CCSD-Inria-SWH collaboration roadmap, there are a few milestones ahead: allowing the deposit of metadata with a link to a code repository which will be archived in SWH or a direct reference to a SWH artifact with the SWH-ID; exporting BibTeX format with a complete @software entry; exporting other software citation formats (e.g. codemeta.json); improving links between teams, people, articles and data to software deposits; and improving the researchers CV export with software research outputs. We believe that these improvements will encourage researchers to share their software and benefit the research and digital curation communities.

[15] https://www.force11.org/group/software-citation-implementation-working-group
[16] https://www.rd-alliance.org/groups/software-source-code-ig
[17] https://gitlab.inria.fr/gt-sw-citation/BibTeX-sw-entry

Acknowledgements

This work is partially supported by the EU Project FAIRsFAIR, call H2020-INFRAEOSC-2018-5-2018-20.

We thank Vicky Steeves, from New York University, for her valuable comments on a preliminary version of this article.

References

Jean-François Abramatic, Roberto Di Cosmo, and Stefano Zacchiroli. (2018). Building the universal archive of source code. Communications of the ACM, 61(10), 29-31. DOI: https://doi.org/10.1145/3183558

Alliez, P., Di Cosmo, R., Guedj, B., Girault, A., Hacid, M. S., Legrand, A., & Rougier, N. P. (2019). Attributing and Referencing (Research) Software: Best Practices and Outlook from Inria. Computing in Science & Engineering, IEEE, In press, pp.1-14. ⟨http://doi.org/10.1109/MCSE.2019.2949413⟩. ⟨hal-02135891v2⟩

ArXiv moderators (2019). Our Moderation Process. In arXiv.org blog. Retrieved on December 6th 2019 from https://blogs.cornell.edu/arxiv/2019/08/29/our-moderation-process/

Barborini, Y., Di Cosmo, R., Dumont, A. R., Gruenpeter, M., Marmol, B., Monteil, A., Sadowska, J., & Zacchiroli, S. (2018). The creation of a new type of scientific deposit: Software. In RDA Eleventh Plenary Meeting, Berlin, Germany. Retrieved from https://hal.archives-ouvertes.fr/hal-01738741

CCSD Development team (2017). Documentation API-HAL: Import SWORD. Retrieved on December 6th 2019 from https://api.archives-ouvertes.fr/docs/sword

Di Cosmo, R. (2019). How to use Software Heritage for archiving and referencing your source code: guidelines and walkthrough. ⟨hal-02263344⟩, see also https://www.softwareheritage.org/save-and-reference-research-software/

Di Cosmo, R., & Zacchiroli, S. (2017). Software heritage: why and how to preserve software source code. In iPRES 2017-14th International Conference on Digital Preservation Sep 2017, Kyoto, Japan (pp. 1-10). Retrieved from https://hal.archives-ouvertes.fr/hal-01590958/

Di Cosmo, R., Gruenpeter M. & Zacchiroli, S. (2018). Identifiers for Digital Objects: the Case of Software Source Code Preservation. . In iPRES 2018-15th International Conference on Digital Preservation, Sep 2018, Boston, United States (pp. 1-9). ⟨10.17605/OSF.IO/KDE56⟩. ⟨hal-01865790v4⟩

Gruenpeter, M. & Sadowska, J. (2018a). Create software deposit: User guide and best practices. (Technical Report). Inria; CCSD; Software Heritage. Retrieved from https://hal.archives-ouvertes.fr/hal-01872189

Gruenpeter, M. & Sadowska, J. (2018b). Moderate software deposit A guide and best practices for the digital archivist.(Technical Report). Inria; CCSD; Software Heritage. Retrieved from https://hal.inria.fr/hal-01876705

IPOL team, Information Processing On Line policy. Retrieved from https://www.ipol.im/meta/policy/ on December 2019

UNESCO-Inria expert meeting (2019). Paris Call: Software Source Code as Heritage for Sustainable Development Retrieved from https://unesdoc.unesco.org/ark:/48223/pf0000366715.locale=en

Jones R. & Lewis.S (2013). SWORD 2.0 Profile. Retrieved on December 6th 2019 from https://web.archive.org/web/20191015204612/http://swordapp.github.io/SWORDv2-Profile/SWORDProfile.html

JOSS team (2019). JOSS review criteria. Retrieved on December 6th 2019 from https://joss.readthedocs.io/en/latest/review_criteria.html also available swh:1:cnt:bcce0f89bd9a8e70e63a4d2d67e99b3cfb1f9d8f;origin=https://github.com/openjournals/joss

Kime P., Wemheuer M., Lehman P., (2019). The biblatex Package Programmable Bibliographies and Citations Specifications (Version 3.13) http://mirrors.ibiblio.org/CTAN/macros/latex/exptl/biblatex/doc/biblatex.pdf

Michael J. (2018). Software Deposit: Guidance for Researchers (Version 1.0). Zenodo. http://doi.org/10.5281/zenodo.1327310

Milliken G.(2019). Self-Archiving Software in Institutional Repositories: Identifying Problems and Proposed Solutions. In IASGE project blog. Retrieved on December 6th 2019 from https://investigating-archiving-git.gitlab.io/updates/Self-Archiving-Software-in-IRs/

Raymond E. S. (2000). Software Release Practice HOWTO. Retrieved on December 6th 2019 from https://www.tldp.org/HOWTO/html_single/Software-Release-Practice-HOWTO/

Smith et al. (2016). Software citation principles. PeerJ Com-put. Sci. 2:e86; http://doi.org/10.7717/peerj-cs.862

Smith AM, Niemeyer KE, Katz DS, Barba LA, Githinji G, Gymrek M, Huff KD, Madan CR, Cabunoc Mayes A, Moerman KM, Prins P, Ram K, Rokem A, Teal TK, Valls Guimera R & Vanderplas JT. (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

Software Heritage Development team (2017). Software Heritage - Deposit: API specifications. Retrieved on December 6th 2019 from https://docs.softwareheritage.org/devel/swh-deposit/index.html