

Beyond FAIR Data: Making Research Software a First-Class Research Output

Roberto Di Cosmo

Director, Software Heritage
Inria and Université Paris Cité

11 June 2026



Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions

Short Bio: Roberto Di Cosmo

Computer Science professor in Paris, now working at INRIA

- 35+ years of research (Theor. CS, Programming, Software Engineering, Erdos #: 3)
- 25+ years of Free and Open Source Software
- 15+ years building and directing structures for the common good



1999 *DemoLinux* – first live GNU/Linux distro

2007 *Free Software Thematic Group*

150 members 40 projects 200Me

2008 *Mancoosi project* www.mancoosi.org

2010 *IRILL* www.irill.org

2015 *Software Heritage* at INRIA

2018 *National Committee for Open Science*, France

2021 *EOSC Task Force on Infrastructures for Software*,
European Union

COMMENT | 07 October 2025

Stop treating code like an afterthought: record, share and value it

Scientists, research institutions, funders, libraries and publishers must all improve software practices.

By [Roberto Di Cosmo](#), [Sabrina Granger](#), [Konrad Hinsén](#), [Nicolas Jullien](#), [Daniel Le Berre](#), [Violaine Louvet](#), [Camille Maumer](#), [Clémentine Maurice](#), [Raphael Monat](#) & [Nicolas P. Rougier](#) 



Illustration: Phil Wheeler

A *Nature* call to action

Nature commentary, October 2025

[doi:10.1038/d41586-025-03196-0](https://doi.org/10.1038/d41586-025-03196-0)

This course is the *how-to* behind that call: how to **archive**, **reference**, **describe**, **cite** and **credit** research software — concretely, and today.

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions

Why Open Science?

Open Science (Second National Plan for Open Science, France, 2021)



Unhindered dissemination of results, methods and products from scientific research.

It draws on *the opportunity provided by recent digital progress* to develop *open access to publications* and – as much as possible – *data, source code and research methods*.

Jean-Eric Paquet (former EU DGRI, on the objective of Open Science)



“Increase *scientific quality*, the *pace of discovery and technological development*, as well as *societal trust in science*.”

Mariya Gabriel (former EU Commissioner for Research)



The COVID-19 crisis has also shown that cooperation at international level in research and innovation is more important than ever, including through *open access to data and results*. *No nation, no country can tackle any of these global challenges alone*.

Yuval Noah Harari (on COVID 19)

“*The real antidote [to epidemic] is scientific knowledge and global cooperation.*”

How we built our scientific knowledge

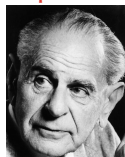
The experimental method



- make an *observation*
- formulate an *hypothesis*
- set up an **experiment**
- elaborate a *theory*

And then we **reproduce** and **verify**.

Reproducibility is the key



non-reproducible single occurrences are of no significance to science

Karl Popper, The Logic of Scientific Discovery, 1934

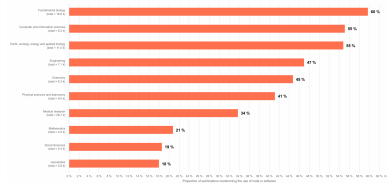
Software is a pillar of Open Science

Software powers modern research

Proportion of publications in France that mention the use of code or software by discipline

Sort by:

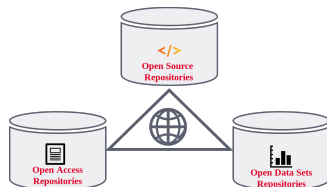
☐ Highest volume ☒ Highest use rate



Over 20% of articles using software across all disciplines share it

2024 French Open Science Monitor

Key pillar: software



Links are **important**

Nota Bene

software may be a *tool*, a *research outcome* and a *research object*

access to the *source code* is essential!

Preserving (the history of) source code is necessary for *reproducibility*



"The source code for a work means the preferred form of the work for making modifications to it."
— GPL Licence

Hello World

Program (excerpt of binary)

```
4004e6: 55
4004e7: 48 89 e5
4004ea: bf 84 05 40 00
4004ef: b8 00 00 00 00
4004f4: e8 c7 fe ff ff
4004f9: 90
4004fa: 5d
```

Program (source code)

```
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

Software Source Code is Precious Knowledge

Harold Abelson, Structure and Interpretation of Computer Programs (1st ed.) 1985

“Programs must be written for people to read, and only incidentally for machines to execute.”

Apollo 11 source code (excerpt)

```
P63SP0T3      CA      BIT6      # IS THE LR ANTENNA IN POSITION 1 YET
              EXTEND
              RAND      CHAN33
              EXTEND
              BZF      P63SP0T4      # BRANCH IF ANTENNA ALREADY IN POSITION 1

              CAF      CODE500      # ASTRONAUT: PLEASE CRANK THE
              TC      BANKCALL      # SILLY THING AROUND
              CADR      GOPERF1
              TCF      GOTOP00H      # TERMINATE
              TCF      P63SP0T3      # PROCEED SEE IF HE'S LYING

P63SP0T4      TC      BANKCALL      # ENTER INITIALIZE LANDING RADAR
              CADR      SETPOS1

              TC      POSTJUMP      # OFF TO SEE THE WIZARD ...
              CADR      BURNBABY
```

Quake III source code (excerpt)

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y; // evil floating point bit level hacking
    i = 0x5f3759df - ( i >> 1 ); // what the fuck?
    y = * ( float * ) &i;
    y = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
    // y = y * ( threehalfs - ( x2 * y * y ) ); // 2nd iteration, this
    // can be removed

    return y;
}
```

Len Shustek, Computer History Museum

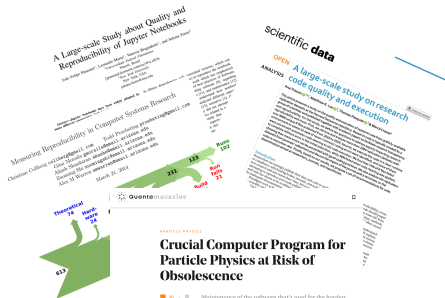
2006

“Source code provides a view into the mind of the designer.”

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions

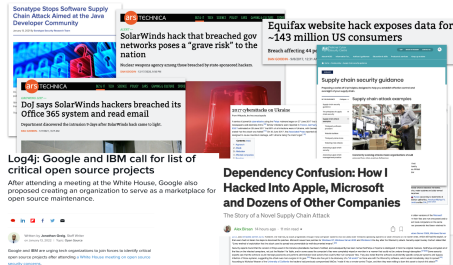
How are we managing our software ?

Reproducibility, maintenance in Academia



(articles: [here](#), [here](#), [here](#) and [here](#))

Security, integrity, traceability in Industry



Can they track the software that they

- ship, use, acquire
- has that bug or vulnerability

Collberg's report from the trenches

Analysis of 613 papers

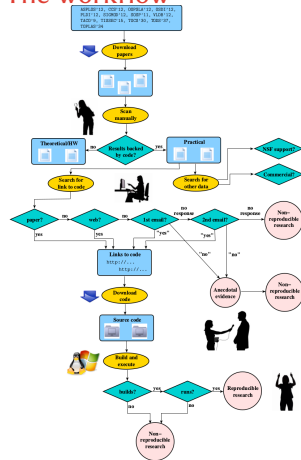
- 8 ACM conferences: ASPLOS'12, CCS'12, OOPSLA'12, OSDI'12, PLDI'12, SIGMOD'12, SOSP'11, VLDB'12
- 5 journals: TACO'9, TISSEC'15, TOCS'30, TODS'37, TOPLAS'34

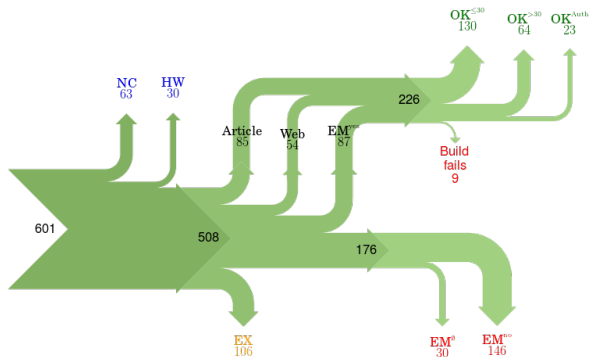
all very practical oriented

The basic question

can we get the code to build and run?

The workflow





... that's a whopping 40% of **non reproducible** works!

Even higher expectations, and yet similarly disappointing results <http://fr.slideshare.net/carloghezzi18/icse-2009-keynote-15919951>

Reference journal

ACM Transactions on Software Engineering and Methodology (TOSEM)

- analysis by Carlo Ghezzi, in 2009, of TOSEM from 2001 to 2006
- 60% of papers refer to a tool
- 20% only are *installable*

Reference conference

International Conference on Software Engineering (ICSE)

- analysis by Zannier, Melrik, Maurer 2006
- complete absence of replication studies

Some people claim that having (all) the source of the code used in an experiment is *not worth the effort* (see “Replicability is not Reproducibility: Nor is it Good Science”, Chris Drummond, ICML 2009)

Sure, diversity *is* important, but:

- Source code is like the proof used in a theorem: can we really accept *Fermat statements* like “the details are omitted due to lack of space”?
- modern complex systems makes even the simplest experiment depend on a wealth of components and configuration options
- access to *all* the source code is not just necessary to *reproduce*, it is also useful to *evolve and modify*, to *build new experiments* from the old ones

Pressure to make research code available is now raising

Evaluation of software artefacts

from optional contest to standard practice



- Started as contest (ESEC/FSE 2011, winner *Vouillon and Di Cosmo*)
- Now a near-universal standard: [ACM Artifact Review and Badging v1.1](#) (5 badges, *left*)

Adopted across computer science communities:

- **PL/SE:** POPL, PLDI, OOPSLA, ICFP, ECOOP, ICSE, FSE, MSR, ISSTA, CAV
- **Systems:** OSDI, SOSP, NSDI, EuroSys, ASPLOS, MICRO, ISCA, HPCA
- **Security:** USENIX Security (availability mandatory in 2025), CCS, NDSS, IEEE S/P
- **HPC and DB:** SC ([Repro. Initiative](#)), SIGMOD ([ARI](#)), VLDB

Best-of-breed exemplars:

- **I POL:** peer-reviewed papers ship code + runnable in-browser demo ([e.g. non-local-means denoising](#))
- **Graphics Replicability Stamp:** 478 graphics papers independently reproduced

EuroSys 2025: 93% of papers earned "Functional"; ~47% pursued "Results Reproduced"

... and now spreading beyond computer science

Economics and social sciences

- [AEA Data Editor](#) (Vilhuber, Cornell): mandatory code+data audit for all AEA journals (AER, AEJ family), since 2019
- [cascad](#): non-profit certification agency (CNRS, HEC Paris, GENES, X); warrants numerical reproducibility, 170+ certifications

Life sciences, geography, and cross-disciplinary

- [CodeCheck](#) (Eglen, Nüst): independent execution of paper code; partner journals incl. GigaScience, Lifecycle Journal, Reproducible AGILE
- [ReScience C](#): open peer-reviewed journal dedicated to replications of computational research

A pan-disciplinary standard

- [NISO RP-31:2021](#): extends ACM-style badging to all sciences

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects**
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions

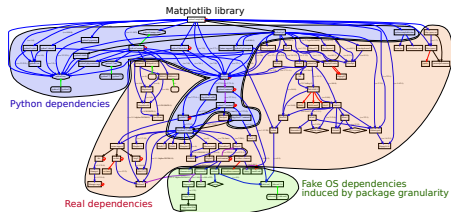
Software source code is *not* data (and FAIR is not the answer)

Software *evolves* over time

- projects may last decades
- the *development history* is key to its *understanding*

Complexity

- *millions* of lines of code
- large *web of dependencies*
 - easy to break, difficult to maintain
 - *research software* a thin top layer
- sophisticated *developer communities*



The human side

design, algorithm, code, test, documentation, community, funding
and so many more facets ...

[Berne Convention](#) (181 states): copyright arises *automatically* on creation — no registration, no notice, no © symbol required.

Default everywhere: "all rights reserved". Collaborators, reusers, industry partners cannot legally touch your code without an explicit *licence* granting them rights.

- The practical menu:
 - **permissive** (MIT, BSD, Apache-2.0): welcomes reuse, no obligation to give back
 - **copyleft** (GPL, LGPL, AGPL): share-alike — modified versions stay open
- Declare it *machine-readably* with an [SPDX identifier](#) in every source file:
SPDX-License-Identifier: MIT
- Automate compliance with the [REUSE Specification](#) (FSFE) and its `reuse lint` CLI
- Decide with [choosealicense.com](#) (60-second guided choice), cross-check [OSI list](#)

Killer fact: ~90% public GitHub repos *have no licence* — block legal reuse despite open intent

Warning: Creative Commons licences are *not* designed for software (no patent terms, no source-vs-binary distinction)

Everybody (machines too!) wants to learn quickly what a code does, make it easy.

- `README.md` — the human entry point: *what, why, install, run, cite, licence*
- `codemeta.json` — rich JSON-LD vocabulary (schema.org-based) for full machine discovery (contributors, dependencies, funding, archival ids); also drives citation via Software Heritage across all major forges — GitHub, GitLab instances, Codeberg, BitBucket, ... (codemeta.github.io)
- Fill it via a web form, not by hand: the [CodeMeta Generator](#)
- `CITATION.cff` — lightweight YAML for citing software; only GitHub renders it as a "Cite this repository" button (citation-file-format.github.io), less powerful than the Software Heritage citation; can be generated from a `codemeta.json` file
- *FAIR Software Five Recommendations* (fair-software.eu): licence + registry + citation + checklist + quality — the de-facto baseline
- *Start small*: README today (10 minutes); add `codemeta.json` once your project grows; generate `CITATION.cff` from CodeMeta

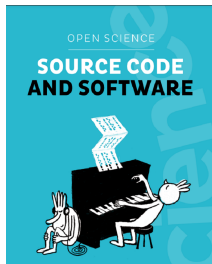
No more "gosh I forgot to add this dependency/configuration/file/image..."

- *A real build system* (not hand-run notebooks): `Makefile`, `cargo`, `poetry`, `cmake`, `npm` — declare the steps, the tool ensures correct order
- *Scientific workflows*: [Snakemake](#), [Nextflow](#), `CWL` — chain compute across cores, clusters, cloud; built-in checkpointing
- *Pin the environment*: `conda`, `pixi`, `uv`, [Nix](#) or [Guix](#), `Docker`, [Apptainer](#) — version lock kills "it worked yesterday"
- *Continuous Integration*: [GitHub Actions](#), [GitLab CI](#) — every push runs tests, builds, optionally publishes
- *Executable papers*: [BinderHub](#) + `Jupyter` — readers re-execute analyses in the browser
- *Aim higher*: [reproducible-builds.org](#) — bit-for-bit deterministic builds, so anyone can verify

No more "myproject-works.py/myproject-nowitworks.py/myproject-save.py/myproject-itusedtowork.py/"

- *Use a modern version control system*, git has >90% adoption ([Stack Overflow 2024](#)); CRAN, PyPI, npm, conda-forge all expect git workflows
- *Host on a forge*: GitHub, GitLab, [Codeberg](#), [SourceHut](#), [gitlab.of.your.institution.or.research.center](#) — git gives you history; the forge adds sharing, issues, CI, discoverability
- *Commit with intent*: small atomic commits, *meaningful* messages; write for future-you ([Conventional Commits](#) is a popular convention)
- *Tag releases* following [semantic versioning](#) (v1 . 2 . 3); tagged releases auto-archive to Zenodo and Software Heritage
- *Branches* + pull/merge requests for review
- *git is not a backup*: pair the forge with [Software Heritage](#) for long-term preservation
- *Go deeper: Pro Git* (free book) at [git-scm.com/book](#)

A passport for source code in Open Science



- ouvrirlascience.fr/source-code-and-software — the French national Open Science booklet on source code (licensing, archival, citation; for students, teachers, researchers)
- [The Turing Way](#) — open handbook on reproducible, ethical, collaborative data science (300+ contributors)
- fair-software.eu — the FAIR-Software Five Recommendations checklist
- reuse.software — machine-readable licensing made painless
- softwareheritage.org/how-to-archive-reference-code — the practical guide (next section)



- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework**
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions

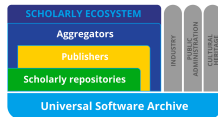
Paris Call on Software Source code (2019, UNESCO)

40 international experts call to “promote software development as a valuable research activity, and research software as a key enabler for Open Science/Open Research, [...] recognising in the careers of academics their contributions to high quality software development, in all their forms”



Open Source in UNESCO [recommendations](#) for Open Science, 2021

Software in the EOSC



2020 [EOSC SIRS](#) connect scholarly ecosystem via Software Heritage

2021 [EOSC Task Force](#) on Infrastructures for Research Software

2022 [FAIRCORE4EOSC project](#) WP6 implements SIRS report

And much more

Software track in [OSEC 2022](#), Software working group launched in Science Europe,
DFG adds software [to model CV \(9/22\)](#), NASA unveils [Open Science policy \(12/22\)](#), ...

French National plan for Open Science, 2021-2024

SECOND FRENCH PLAN FOR OPEN SCIENCE

Generalising open science in France 2021-2024



Second French Plan for Open Science



Launch on 6 July 2021 by Frédérique Vidal, Minister for Higher Education, Research and Innovation

- Multiplying the **levers for change** in order to **generalise open science practices**
- Structuring the **policy for opening up or sharing research data**
- New commitments to the **opening of source code** produced by research
- **European and international inclusion** in the context of the French Presidency of the European Union
- **Disciplinary and thematic variations**: open science policies must be adapted to disciplinary specificities

Path Three : Opening up and promoting source code produced by research

7

Recognize and support the dissemination under an open source license of software produced by publicly funded research programmes

« The opening of software source code is a major challenge for the **reproducibility** of scientific results. »

8

Highlight the production of **source code** from higher education, research and innovation

« Distribution of software products under **open source licence** will be preferred. »

9

Define and promote an **open source software policy**

Define and promote an open source software policy

- Produce a **National Charter for Open Source Software** coming from higher education, research and innovation
- Develop the **link between data and software** through a network of **Chief Data Officers** in the various universities and research performing organisations.
- Develop the **economic models of open source software** and make them known within commercialization services
- **Support Software Heritage** and recommend it for the archiving and referencing of source code

Recognise source code as a contribution to research

- Create an **open source research software prize**
- **Provide greater recognition** for software production in the career of researchers, research support staff

Build an ecosystem that connects code, data and publications

- Develop **proper coordination** between software forges, open publication archives, data repositories and the scientific publishing sector.

Five action lines (see [details online](#))

- Identifying and highlighting research software production
- Technical and social tools and best practices
- Valorization and sustainability
- Liaison and animation at national, European, and international levels
- Recognition and careers

Leveraging experience and connections

- Open Source thematic group in Systematic (since 2007, more on demand)
- Collaboration with DINUM, Eclipse Foundation, OW2, ...

Co-chairs

- *Roberto Di Cosmo* – Professor, Inria and Université Paris Cité; Director, Software Heritage
- *François Pellegrini* – Professor, Université de Bordeaux; member of the French data-protection authority (CNIL)

40+ active members in five working groups (GT1–GT5)

- **GT1** identify and highlight research software *L. Albaret, V. Louvet*
- **GT2** tools and best practices *J.-Y. Jeannas, J.-C. Souplet*
- **GT3** valorization and sustainability *K. Schmidmayer, S. Tonda-Goldstein*
- **GT4** national / European / international liaison *K. Hinsén, R. Monat*
- **GT5** recognition and careers *M. Giraud, C. Marmo*

members from Inria, CNRS, IRD, Cirad, CEA and many universities

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them**
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions

Source code in Open Science: a plurality of needs to address

Researchers

- archive and reference software used in articles
- find useful software
- get credit for developed software
- verify, reproduce, improve results

Laboratories/teams

- track software contributions
- produce reports
- maintain web page

Research Organizations and/or Funders

know its software assets

- technology transfer
- funding strategy
- impact metrics
- career evaluation

ARDC

- **Archive** for retrieval (*reproducibility*)
- **Reference** for identification (*reproducibility*)
- **Describe** for discovery and reuse
- **Cite/Credit** for credit and evaluation

Before ARDC

- **Development** practices and tools (VCS, build system, test suites, CI, code quality, ...)
- **Opening up** towards a community (documentation, organization, communication)

Need training, tooling, infrastructures, best practices

Beyond ARDC

- **Policies** (dissemination, reuse, careers, ...)
- **Sustainability** (legal, financial, etc.)
- Technology transfer
- Advanced technologies and tools (quality, traceability, etc.)

Here we will focus on ARDC

Archive

Research software artifacts must be properly **archived**
make sure we can *retrieve* them (*reproducibility*)

Reference

Research software artifacts must be properly **referenced**
make sure we can *identify* them (*reproducibility*)

Describe

Research software artifacts must be properly **described**
make it easy to *discover* and *reuse* them (*visibility*)

Cite/Credit

Research software artifacts must be properly **cited** (*not the same as referenced!*)
to give *credit* to authors (*evaluation!*)

Archive and reference: some popular approaches that do not fit the bill

A - Since the 1970's 1990's

.zip or .tar file on:

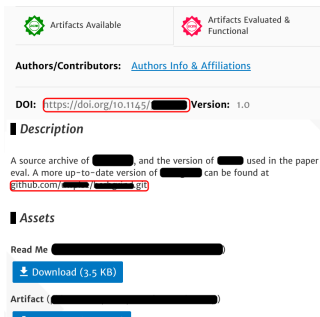
- ftp-server (e.g. [gnu](#))
- web page ([example](#))
- document archive (+ DOI [sample](#))

B - Since the 2000's

Rely on *software forges*

- institutional/project (e.g. [example](#))
- free commercial ones: BitBucket, GitHub, GitLab, ... (e.g. [parmap](#))

C: a mix of the two



Can get no satisfaction...

A *Poor user experience*

B *No preservation guarantee*

C *Can do so much better*

Where is the source code?

Collaborative development platforms (aka "forges")

- BitBucket, GitLab(.com), GitHub, etc.
- support for version control, issues, etc.
- example:
 - <https://github.com/rdicosmo/parmap>
 - <https://gitlab.inria.fr/gt-sw-citation/bibtex-sw-entry/>

Distribution platforms

- CTAN, CRAN, PyPi, Debian, etc.
- example: <https://ctan.org/pkg/biblatex-software>

Archives

- Software Heritage
- example: [archived version of biblatex-software](#)

Forges are *not* archives!

2015: the first big bad news

Google Code and Gitorious.org shutdown: ~1M endangered repositories

- broken links in the web of knowledge (my papers too)

Big bad news keep coming in

- summer 2019: BitBucket announces Mercurial VCS sunset
- july 2020: BitBucket erases 250.000+ repositories (including research software)
- summer 2022: GitLab.com considers erasing **all** projects that are **inactive for a year**

In Academia too!

- 2021: Inria's old gforge is unplugged... **breaks the Opam build chain** for OCaml

We need a universal archive of software source code: now we have one!

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code**
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Collect, preserve and share *all* software source code

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

Reference catalog



find and reference all
software source code

Universal archive



preserve and share all
software source code

Research infrastructure



enable analysis of all
software source code

Sharing the vision



And many more ...

www.softwareheritage.org/support/testimonials

Donors, members, sponsors



Diamond sponsors



Platinum sponsors



Gold sponsors



Silver sponsors



Bronze sponsors



The largest software archive, a shared infrastructure

Cultural Heritage



Industry



Research



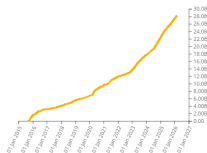
Public Administration



Software Heritage

Source files

28,178,230,711

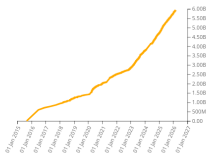


Directories

22,013,760,256

Commits

5,925,711,720

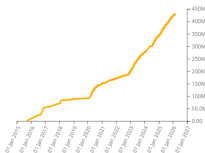


Authors

105,360,419

Projects

430,218,860



Releases

147,686,240

Bitbucket

1,925,997 origins

debian

128,719 origins

GitLab

3,982,586 origins

heptapod

1,068 origins

NixOS

12,032 origins

Phabricator

192 origins

git

21,603 origins

Guix

5,947 origins

launchpad

329,908 origins

npm

1,799,296 origins

python

410,582 origins

R

21,113 origins

GitHub

137,564,899 origins

GNU

354 origins

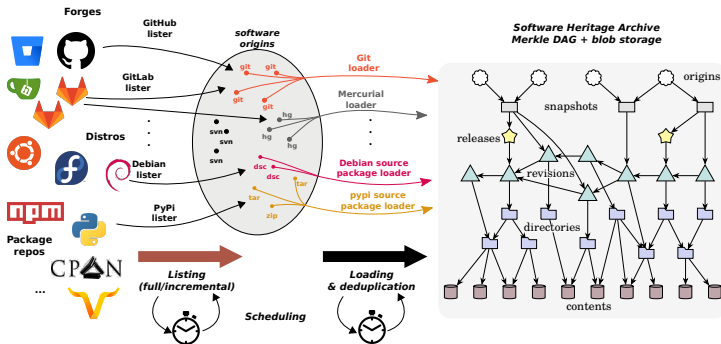
Maven

93,738 origins

SourceForge

308,990 origins

Address common Open Science and Open Source needs: archival



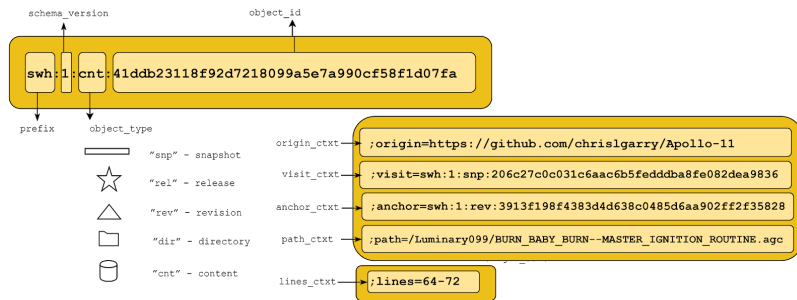
Global development history **permanently archived** in a **uniform data model**

- over **28 billion** unique source files from over **430 million** software projects
- **~3PB** (compressed) blobs, **~60 B** nodes, **~1 trillion** edges

Address common Open Science and Open Source needs: reference

Meet the SWHID identifier

Software Hash Identifiers (SWHID)



[link to full docs](#)

25+B

intrinsic,
decentralised,
cryptographic

Full fledged *source code references* for traceability, integrity and reproducibility

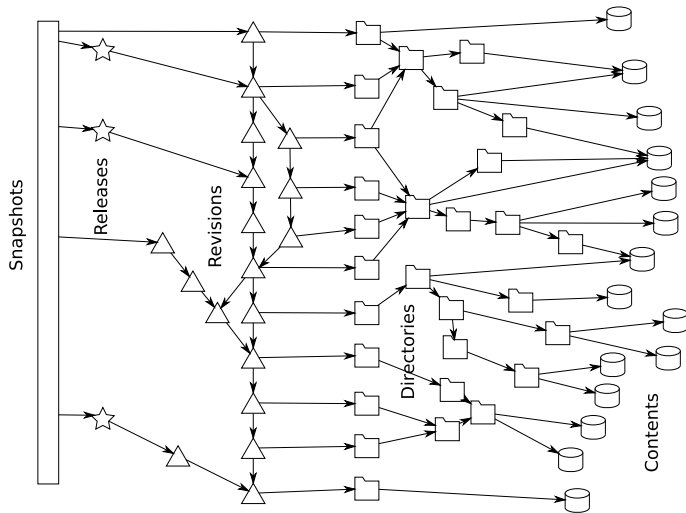
Examples: [Apollo 11 AGC excerpt](#), [Quake III rsqrt](#)
Guidelines available, see [the HOWTO](#)

- Linux Foundation [SPDX 2.2](#)
- IANA-registered "swh:"
- WikiData property [P6138](#)

Now an ISO standard: [ISO/IEC 18670:2025](#)
implementation: [swh.model](#)

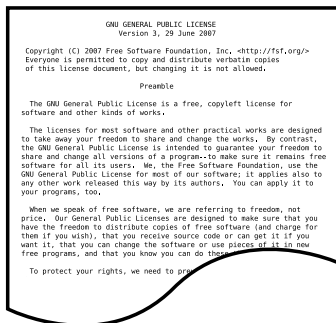
ref.

How a SWHID is built: a Merkle DAG of the development history



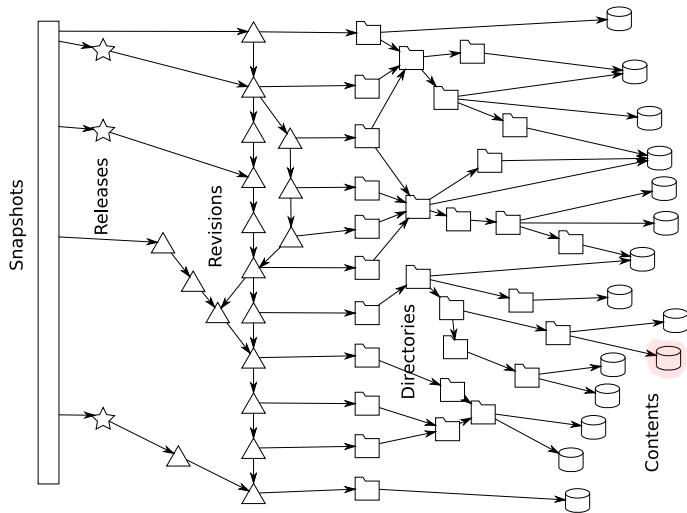
How a SWHID is built: a Merkle DAG of the development history

Contents



sha1: 8624bcdade55baeef...
sha256: 8ceb4b9ee5aded...
sha1_git: **94a9ed024d385...**
length: 35147

How a SWHID is built: a Merkle DAG of the development history



How a SWHID is built: a Merkle DAG of the development history

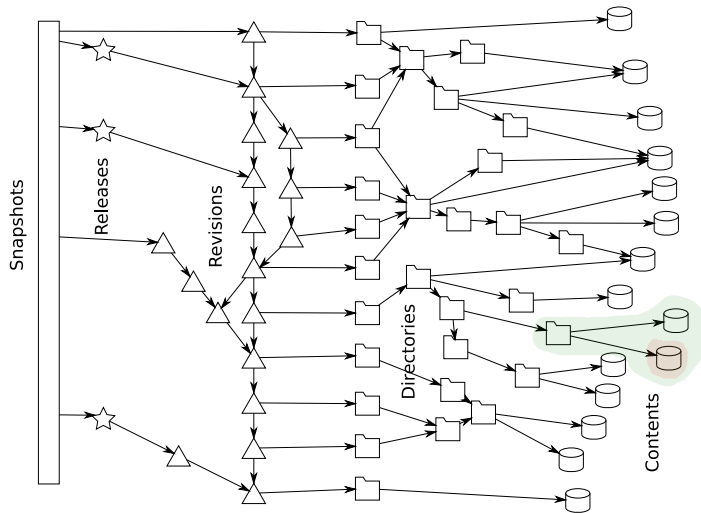


Directories

```
100644 blob c5baade4c44766042186ef858c0fd63d587ebf09 .gitignore
100644 blob 2d0a34af6f52cf3cf6b0c2f7bd0648fbd255e77f AUTHORS
100644 blob 94a9ed024d3859793618152ea559a168bbcbb5e2 LICENSE
100644 blob d9b2665a435a43f8a79a84e0867751dfb095c7bb MANIFEST.in
100644 blob 524175c2bad0b35b975f79284c2f5a6d5eaf2eb4 Makefile
100644 blob 5c7e3a5bbddb038682ba7793f440492ed9678bb3 Makefile.local
100644 blob 8617980629cd24e6080404f09aa749b085b3e07b README.db_testing
100644 blob 76b29f94cf815e0869c414d38d78d7ce08ec514e README.dev
040000 tree e1e10ecef948af0b93adb0372afc89f12e92618a bin
040000 tree 83e56d0beaf7793c77a45a345c80fcb8af503013 debian
040000 tree a34c9c4ba213f0cedc67f9816348d27955577af5 docs
100644 blob f2a6d32c6135aa7287bbd76167b01df2ae4f1539 requirements.txt
100755 blob eee147c36caf1bbc2d820da8dc026cb5b68180bc setup.py
040000 tree 224bb4c1f4c67fca1d160bffdd2d06094e7e1abf3 sql
040000 tree 8631c9cd77bbe993168107ab5baf51f40c6300be swh
040000 tree 8fb905b56ba8ed692f1209b2773b474c6c1d66c1 utils
```

id: 515f00d44e92c65322aaa9bf3fa097c00ddb9c7d

How a SWHID is built: a Merkle DAG of the development history



Revisions

Details	Changes	Files
SHA: 963634dca6ba5dc37e3ee426ba091092c267f9f6		
Author: Nicolas Dandrimont <nicolas@dandrimont.eu> (Thu Sep 1 14:26:13 2016)		
Committer: Nicolas Dandrimont <nicolas@dandrimont.eu> (Thu Sep 1 14:26:13 2016)		
Subject: provenance.tasks: add the revision -> origin cache task		
Parent: fc3a8b59ca1df424d860f2c29ab07fee4dc35d10 : test_storage: properly pipeline origin and cont...		
provenance.tasks: add the revision -> origin cache task		
swf/storage/provenance/tasks.py		



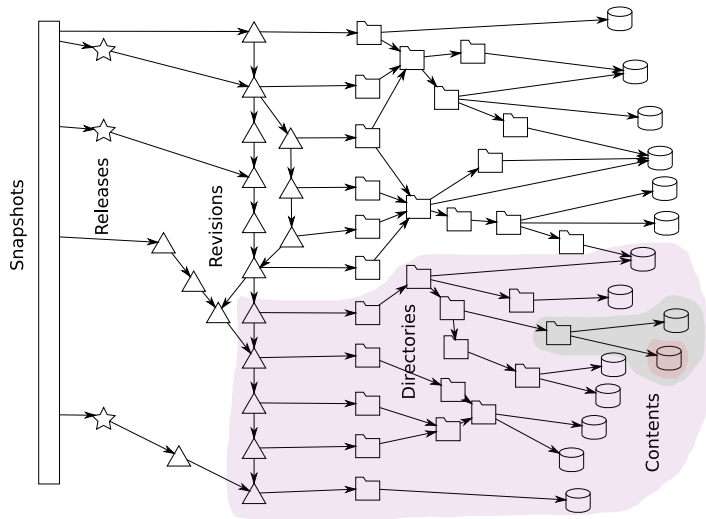
77

tree 515f00d44e92c65322aaa9bf3fa097c00ddb9c7d
parent fc3a8b59ca1df424d860f2c29ab07fee4dc35d10
author Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200
committer Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200

provenance.tasks: add the revision -> origin cache task

id: 963634dca6ba5dc37e3ee426ba091092c267f9f6

How a SWHID is built: a Merkle DAG of the development history



How a SWHID is built: a Merkle DAG of the development history

Releases

tag v0.0.51
Tagger: Nicolas Dandrimont <nicolas@dandrimont.eu>
Date: Wed Aug 24 14:36:03 2016 +0200

Release swh.storage v0.0.51

- Add new metadata column to origin_visit
- Update swh-add-directory script for updated API [...]

commit c0c9f16b1e134f593e7567570a1761b156e6eb1d

object c0c9f16b1e134f593e7567570a1761b156e6eb1d
type commit
tag v0.0.51
tagger Nicolas Dandrimont <nicolas@dandrimont.eu> 1472042163 +0200

Release swh.storage v0.0.51

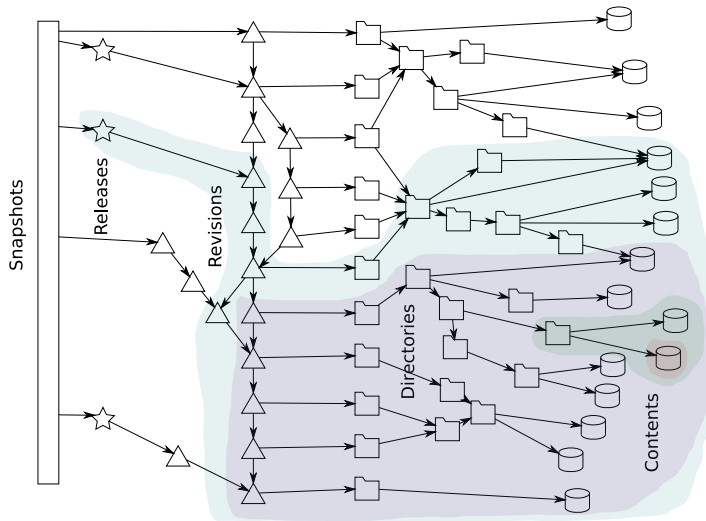
- Add new metadata column to origin_visit
- Update swh-add-directory script for updated API

-----BEGIN PGP SIGNATURE-----

iQIzBAABCAAdBQJXvZTNFhXuaWNvbGFzQGRhbmRyaW1vbnQuZXUACgkQ7AWLMo2+
neqorw/aaq6SOb5DijEa+kWN3rXgV5+1K1vEVh1wNKAwx8eKJ7aX2kEiLDtt7uf
ahpZ6pz3q8nqs6aC1+YrxBfcih3L2YtrdZeWXWqr8xWNMaEoYDb8qaphwh8AD5t2
ICBIt2ujtXuCrDt93eKKPvwZxg+h80sMWy35Dr6jW7Z7K4Mu/PgGlyIHPY55yo
IGEndWno7Vfh1Vm6t1n5qB7l5mXRagA+becqddubTZ2xij+jpIUqC8cyqN3hm/fL
qsJ2mu8kyz3t8tG/H1/pV+I5OwBlNpO55TH0tuoJEvGPK/dHSP79QuHDHZFkCao
kij6kAWyU80Mxb+nKVjLbrR3+yWBFj3Qp5a1/V8oOTd6E1dALcNMpEaKCoKtMt
d/gMRax1I1/g0EDfnsW67G6sDwKPKPhgFVLQ3nV3GaQQTnu1RpMz006H9/tAwzC
Gg/K1PdHT4hz0i46wYPZyje0U2VXGFu6vU9vFQ4ZR/Wjn+0zMzdcRdrIJSUOMn
RpTTFUsbXUeXHGOpkgXhSYTnvp1gdPc76USTsK0aGe84AZm1Ik0mGrwXCVFpqlYo
nhhibB5HBNMoqyF6yTSOpUbyK70tpYRRUGKWDerK0wKSxkWUJGtKzy6jYqJto29
gulwgZQif5qWQCBOontAL2+HvPFaVyckMejUhg62cP/+EHivUk=
=kOxP
-----END PGP SIGNATURE-----

id: 85083a5cc14a441c89dea73f5bdf67c3f9c6afdb

How a SWHID is built: a Merkle DAG of the development history



How a SWHID is built: a Merkle DAG of the development history

Snapshots

git show-refs

```
commit 08ffeb25770109525eb3ce21691466c53a1d9158 refs/heads/atime
commit ba543a24e3f9fe323a46c292cec4fcb61c67eb refs/heads/directory-listing-arrays
commit d69e0dbf892383ff6589b27fbelc05d27238d9c5 refs/heads/foo
commit c7ff9eea0eb22f8946908f5a8019f67de468e08 refs/heads/master
commit 7eca197fc66d2024047e54bled9e8b44361a0fc2 refs/heads/tmp-directory-add
commit 642a205f37de85005a85d427b53ee4fb2252e82e refs/heads/tmp/generic-releases
tag 20f043b1379cf768d966597799fd4907c757f755 refs/tags/v0.0.1
tag 72a21991a384e539996dbb867bfb0bee72aee2cd refs/tags/v0.0.10
tag 3590e0ca0ebb070e5b376705fa230bbfa4ffa5cc refs/tags/v0.0.11
tag 33378427a403ba569a67777b8d58f6674fbc6556 refs/tags/v0.0.12
tag 06f74652755b327cf590311c2bfa036cf3b4b35d refs/tags/v0.0.13
tag 5a6325fe86ab854b581d7442667d92a11e32f3bd refs/tags/v0.0.14
tag 586fba4e580b4f5fab05f599367643cbb1a9c7f refs/tags/v0.0.15
tag 8cd8b885f4098bf363177742bd289f660e5be51c refs/tags/v0.0.16
tag a542444ee3f0fbed35efb202fee035c809abc7d6 refs/tags/v0.0.17
tag 228a2f1650dd12222e556559462e1e06fc4993d9 refs/tags/v0.0.18
tag 606979a4ca05d497fc0d24aad0dce82636ef47c refs/tags/v0.0.19
tag 32bf5a59fc2a323baa6d5f15a6ad5382ec275a67 refs/tags/v0.0.2
tag 3147c3d31ec46cf6492f881e908b1237ebdff2c7 refs/tags/v0.0.20
tag 215ea50daba11e082e0b72e76eb4b6073a87908 refs/tags/v0.0.21
tag 3fb168c2072a5d6252124257a1e5dfc0f5ffa1df refs/tags/v0.0.22
tag 8cdbee8da4d73fc5d262789e460a16ac3c72aba4 refs/tags/v0.0.23
...
```

id: b464cad1b66fff266a37b46ea6e7a04b545e904b

Don't trust, verify: recompute the SWHID yourself

The SWHID is computed from the *content*: anyone can recompute it *offline* with the reference implementation (`swh.model`).

```
$ SWHID=swh:1:dir:HASH
# 1. download that directory as a tarball (the archive "vault")
$ curl -o tree.tar.gz "$ARCHIVE/api/1/vault/flat/$SWHID/raw/"
# 2. unpack it, then 3. recompute its SWHID locally
$ tar xf tree.tar.gz
$ swh identify --type directory HASH/
swh:1:dir:HASH      # identical: no need to trust the archive
```

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!**
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions

A walkthrough

- Browse (e.g. [Apollo 11 \[excerpt\]](#), your work [may be already there !](#))
- Trigger archival, use the [updateswh](#) browser extension, configure the webhooks
- Get and use SWHIDs ([full specification available online](#))
- Cite software with [biblatex-software](#) package from CTAN
 - [Overleaf ACMART template](#) available
- Example in journals: [article from IPOL](#)
- Example with Parmap: [devel on Github](#), [archive in SWH](#), [curated deposit in HAL](#)
- Extracting all the software products [for Inria](#), [for CNRS](#), [for CNES](#), [for LIRMM](#) or [for Rémi Gribonval](#) using [HalTools](#)
- Curated deposit in SWH via HAL, see for example: [LinBox](#), [SLALOM](#), [Givaro](#), [NS2DDV](#), [SumGra](#), [Coq proof](#), ...
- Example use in research articles:
 - compare Fig. 1 and conclusions in [the 2012 version](#) and [the updated version](#)
 - SWHID in [a replication experiment](#)

A flagship life-sciences journal: code preservation built *into the publishing pipeline*, not left to author goodwill.

- *Before 2020*: code was *forked* into the GitHub org [elifesciences-publications](#) — no PID, manual licensing, did not work for non-GitHub repos
- *Since 2020*: every article's code is archived in Software Heritage; the SWHID is embedded in the JATS XML and rendered on the article landing page
- *Validation pipeline*: Schematron rules over the JATS XML *flag* every mention of code for vendors and *verify* that the SWHID matches the source repo before publication
- *SWHID structure example* (population genetics, [Zaidi et al.](#)):
=swh:1:dir:506ed3...;origin=github.com/Arslan-Zaidi/popstructure;anchor=swh:1:rev:1509a5... =
([browse the full qualified SWHID in the archive](#))
- *Researcher view* (P. Poulain et al., [eLife 90061](#), 2024): SWH hyperlinks in the [reviewed preprint](#) are transformed by the editorial team into proper bibliographic references in the [published Version of Record](#)
- Announced 12 November 2020 ([eLife post](#), [SWH announcement](#))

A look at some adoption indicators

From [Melissa Harrison's OSEC 2022 talk](#)



What are they “referencing”?

source	n	percentage
Not available	2868	46.22
GitHub	1151	18.55
software heritage	387	6.24
zenodo	142	2.29
r package	70	1.13
cran	56	0.90
r package version	54	0.87
gitlab	35	0.56

- 6205 “software” references identified
- Top 8 listed, then long tail of 1055 other sites – 932 are unique “source”

Standard on replicabilitystamp.org

Lightweight Curvature Estimation on Point Clouds
with Randomized Corrected Curvature Measures

Jacques-Olivier Lachaud, David Coeurjolly, Céline Labart, Pascal Romon,
Boris Thibert
Wiley Computer Graphics Forum (CGF)



HAL+SWH in the [Open Science software booklet](#)

Funding agencies recommendations [ANR 2023 guidelines](#) (p. 17)

Enfin, conformément au 2^{ème} Plan national pour la science ouverte, L'ANR recommande que les logiciels développés durant le projet soient mis à disposition sous une [licence libre](#)³⁰ et que les codes sources soient stockés dans l'archive Software Heritage³¹ en indiquant la référence au financement ANR.

Software Heritage is a game changer

- **universal** archive
 - **all** the source code
 - yes, even 20 lines snippets/scripts for data mangling
 - more importantly: all the code your code depends upon!
- SWHID: uniform **intrinsic** identifier
 - independent of the development platform or technology
 - full guarantee of integrity (climate data anecdote goes here)

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers**
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions

Archive in one click: the *UpdateSWH* browser extension

Check and refresh the archive from your repository page

- for Firefox, Chrome and Edge; works on GitHub, GitLab, Bitbucket, Gitea, Codeberg
- a colour-coded button shows the archival status of the page you are viewing

What the button does

- **grey** — not archived yet → click to archive it
- **yellow** — outdated → click to trigger an update (Save Code Now)
- **green** — up to date → click to open the archive / copy the SWHID

no account, no setup: install and click

Archive automatically: forge *webhooks*

Preserve every important version with *zero* ongoing effort

Add *one* webhook URL in your repository settings:

`https://archive.softwareheritage.org/api/1/origin/save/webhook/gith`

(also `.../gitlab/` and `.../bitbucket/`)

- recommended: trigger on *tag, branch or release creation* — not on every push
- supported: GitHub, GitLab (and instances), Bitbucket, Gitea, SourceForge

see the [Software Heritage webhooks guide](#)

Describe your software: *CodeMeta* and CITATION.cff

Machine-readable metadata that travels *with* the code

- `codemeta.json` — a JSON-LD schema for software (authors, license, references, funding, ...), Software Heritage uses it for the multiple granularity citation button
richer than CITATION.cff
- `CITATION.cff` — the lightweight cousin; GitHub renders it as a “*Cite this repository*” button, Software Heritage uses it for the multiple granularity citation button
can be generated from CodeMeta

No JSON by hand: the CodeMeta generator

- fill in a web form → download a ready `codemeta.json`
codemeta.github.io/codemeta-generator
- drop both files at the root of your repository → archived *and* indexed by Software Heritage

A dedicated citation style

(in TeXLive; used by ACM templates since 2022)

Four entry types – `@software`, `@softwareversion`, `@softwaremodule`, `@codefragment` – carrying a `swhid` field, chained with `crossref`.

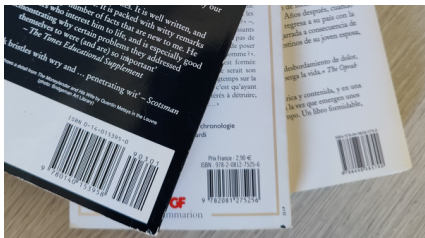
A worked example

```
@softwareversion{parmap-1.1.1,  
  author   = {Di Cosmo, R. and Danelutto, M.},  
  title    = {The Parmap library}, version = {1.1.1}, license = {LGPL-2.0},  
  swhid    =  
{swh:1:rel:373e26...db773;origin=https://github.com/rdicosmo/parmap}}
```

the `swhid` makes the citation *reproducible*, long after URLs break

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion**
- 11 Software Heritage, cont'd
- 12 Actions

Identification of a book



Goal: identify a book

- one ISBN number per published book
- ISO 2108 standard specification

Location of (a copy of) a book



Goal: find (a copy of) a book

- many locations (locations can change!)
- many approaches for call numbers

identification and **location** are *separate concerns*

The Digital Object Identifier (DOI), and its trust assumptions

Example: <https://doi.org/10.1109/MSR.2015.10>

- to find what 10.1109/MSR.2015.10 is, go to a *resolver* (e.g. doi.org)
- this returns <http://ieeexplore.ieee.org/document/7180064/>
- at this URL we find ...

Related Conferences: Mining Software Repositories

Mining Component Repositories for Installability Issues

View Document | 1 Page | 45 Citations | Not Listed

Related Articles: Software documentation for program maintainability | Projecting Software Maintainability from Static Analysis | Software documentation for program maintainability

Authors: Pierre Nadeau, Fabrice Di Cosmo, Louis Gauthier, Fabrice Le Pessier, Yifan Tang, ... | 6 Authors

Abstract: Component repositories play an increasingly relevant role in software life cycle management, from software distribution to end users, to deployment and upgrade management. Software components shipped via such repositories are equipped with rich metadata that describe their relationship (i.e., dependencies and conflicts) with other components. In this practice paper we show how to use a tool, *distcheck*, that uses component metadata to identify all the components in a repository that cannot be installed (i.e., due to unsatisfiable dependencies), provides detailed information to help developers understanding the cause of the problem, and fix it in the repository, via report about detailed analysis of several repositories: the Debian distribution, the CPAN package collection, and Chapel modules. In each case, *distcheck* is able to effectively identify not installable components and provide valuable explanations of the issues. Our experience provides solid ground for generalizing the use of *distcheck* to other component repositories.

Published in: Mining Software Repositories (MSR), 2015 IEEE/ACM Joint Working Conference on

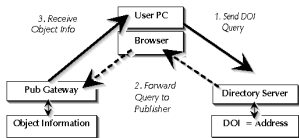
Date of Conference: 20-21 May 2015 | **IEEEPC Acquisition Number:** 13370108

Date Added to IEEE Xplore: 04 August 2015 | **DOI:** 10.1109/MSR.2015.10

Electronic ISBN: 978-0-7691-0104-2 | **Publisher:** IEEE

Download PDF | Read the full document

Architecture of the DOI infrastructure



- DOI resolution *can change*
- content at URL *can change*
- no *intrinsic* way of noticing
- persistence based on *good will of multiple parties*

Extrinsic vs Intrinsic identifiers

In a nutshell

(for more info see [this dedicated blog post](#))

Main difference: how the *relation* between *identifier* and *designated object* is created and maintained. *Persistence* is a key desired property.

	Extrinsic	Intrinsic
relation	register	convention
persistence	external ¹	internal
pre-internet	passport number, ISBN, SSN, etc.	Music/Chemistry notations <i>e.g. NaCl is table salt</i>
internet era	DOI, Handle, Ark, etc.	cryptographic hashes <i>e.g.: git, bitcoin, SWHID</i>

distributed software development (e.g. git) relies on *intrinsic identifiers*
uniform intrinsic identifier

we need a

¹"persistence... is a function of *administrative care*" [RFC 3650 \(Handle System Overview, 2003\)](#)

A DOI pointing to a Zenodo *zip*

- resolves to a **snapshot** you must *trust* — no built-in integrity check
- one opaque blob: no *history*, no *diff*, nothing addressable *inside* it
- granularity stops at "the whole deposit"

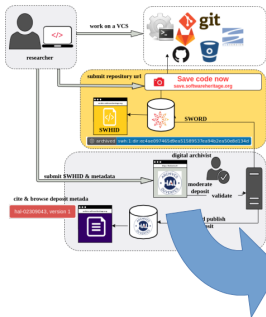
A *SWHID*

- **intrinsic**: the identifier *is* the cryptographic hash → self-verifying
- addresses *any* level: snapshot / revision / directory / file / **lines**
- works for *any* VCS (git, hg, svn) and for plain tarballs

a DOI answers "*which deposit?*"; a SWHID answers "*exactly which bytes, and prove it*"

HAL and Software Heritage: building a curated software catalog

<https://hal.archives-ouvertes.fr/hal-02130801>





Free and accessible knowledge

[Home](#) | [Submit](#) | [Browse](#)- | [Search](#) | [Documentation](#)

LinBox

The LinBox Group : [L I N B O X](#) (Gnu)

- ☐ EDO - Exact Computing
- ☐ UMR - Laboratoire d'Informatique de la Robotique et du Modélisation de Montpellier
- ☐ ARAC - Arithmetic and Computing
- ☐ IGRA - Informatique - Rhône-Alpes, LIP - Laboratoire d'Informatique du Parallélisme
- ☐ AMALCON - Algorithms and Software Architectures for Cloudability and HPC Platforms
- ☐ Inria Grenoble - Rhône-Alpes, LIP - Laboratoire d'Informatique du Parallélisme
- ☐ CGO - Department of Computer and Information Sciences [Newark]
- ☐ Ghent University
- ☐ NCJG - Department of Mathematics [Raleigh]
- ☐ United States Naval Academy
- ☐ SGO - Symbolic Computation Group
- ☐ CASC - Calcul Algébrique (Symbolique), Sécurité, Systèmes Complexes, Codes et Cryptologie

L.I.K. - LinBoxe Jean Kuentzmann

Abstract: LinBox is a C++ template library of routines for solution of linear algorithms problems including linear system solution, rank, determinant, minimal polynomial, characteristic polynomial, and Smith normal form. Algorithms are provided for matrices with integer entries or entries in a finite field. A number of matrix storage types are provided, especially for blocksize representation of sparse or structured matrix classes. A few algorithms for rational matrices are available. LinBox also uses underlying data structures and algorithms for integer, rational, polynomial, finite fields and rings, as well as dense and sparse matrix formats coming from the Givaro (<https://cristal.gforge.inria.fr/~givre/givaro/>) and FFLAS-FFPACK (<http://bitbucket.gforge.inria.fr/repos/fflas/>) libraries.

METADATA

VARIATION

1.6.3

Software License

GML Lesser General Public License v3.0 or later

Programming Language

C++

Code Repository

<https://github.com/linbox-team/linbox>

COLLECTIONS

ENG-LYON | CNRS | LRMM | ECO - LYON | LYON - ILM | IRISA | IMAG-MONTPELLIER | LK | LUK_MAD | LUK-MAD-CASG | ULY | UGA | ANR

CITATION

The LinBox Group. LinBox. 2019. (vwh-1r-39300) 61n4xM4Q3d3k366H8W5D3Z3T1vU8tjg.org/<https://arxiv.org/archive/other/2019/07/2019071401.pdf> new+1: <https://doi.org/10.26907/26906/26292/7016a8bbf0a2451ad411c1b4damp117ee4de15C392E29B978754629636119E3b1406172.pdf?v=1>. (hal-02130001)

Document type **[[Enhance](#)]**

Datasets

Computer Science [en]

Computer Science [fr] • Symbolic Computation [en SG]

Complete list of metadata

[Display](#)

BROWSE

Software Engineering

wvh-1r-39300/1141242832d3598b767253271c1b5e1.org/en?hl=en&search=hal%20linbox&sortby=date&filterby=tag%3A%20linbox&view=list <http://1952928809/02623a707a1d4610742451ad411c1b4damp117ee4de15C392E29B978754629636119E3b1406172.pdf?v=1>

[Browse](#)

Enter a SWHID to resolve or keyword(s) to search for it

🔍

Search

📄

https://hal.archives-ouvertes.fr/hal-02130801 🔗

🕒

14 June 2019, 13:43 UTC

< Code

🌿 Branches (t)

📁 Releases (o)

📂 Vists

Revision ebf818328952266b7875c69296b1963b1496107 ✖

393b6b1 / linux-1.6.3 / linux / config-blas.h

Download File 📄

Tip revision: ebf818328952266b7875c69296b1963b1496107 authored by Software Heritage on 11 June 2019, 08:12 UTC
hal: Deposit 297 in collection hal

config-blas.h

```
1 /* config-blas.h
2 * Copyright (C) 2005 Pascal Giorgi
3 *      2007 Clement Pernet
4 * Written by Pascal Giorgi <pgiorgi@waterloo.ca>
5 *
6 * =====LICENCE=====
7 * This file is part of the library Linbox.
8 *
9 * Linbox is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public
11 * License as published by the Free Software Foundation; either
12 * version 2.1 of the License, or (at your option) any later version.
13 *
14 * This library is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
17 * Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public
20 * License along with this library; if not, write to the Free Software
21 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02118-1301 USA
22 * =====LICENCE=====
```

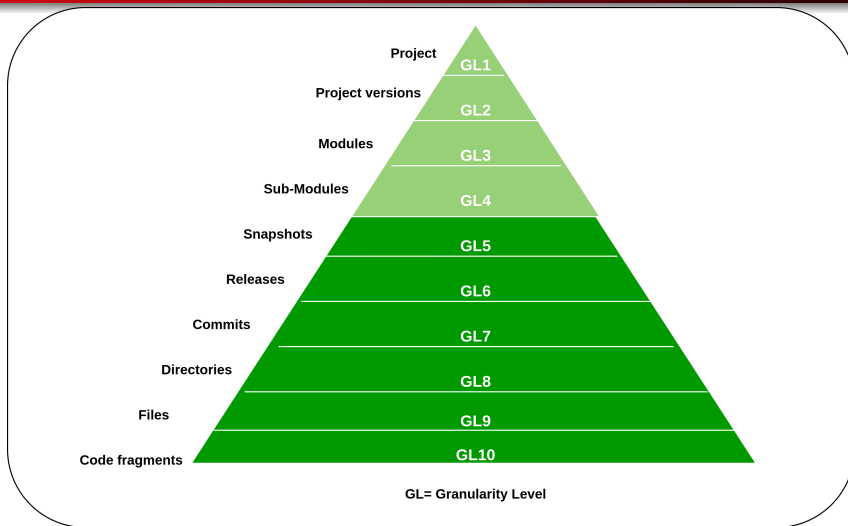
swh:1:dir:393b611a1424f032e83569bf6762502371cfcf65

with minimal user overhead!

(CC-BY 4.0) Beyond FAIR Data 11/6/2026

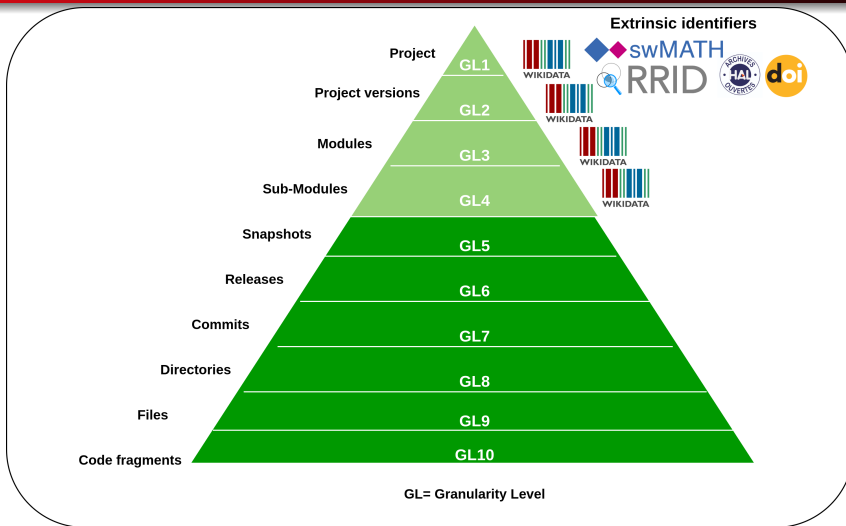
52 / 59

Under the hood: a uniform data model with intrinsic identifiers



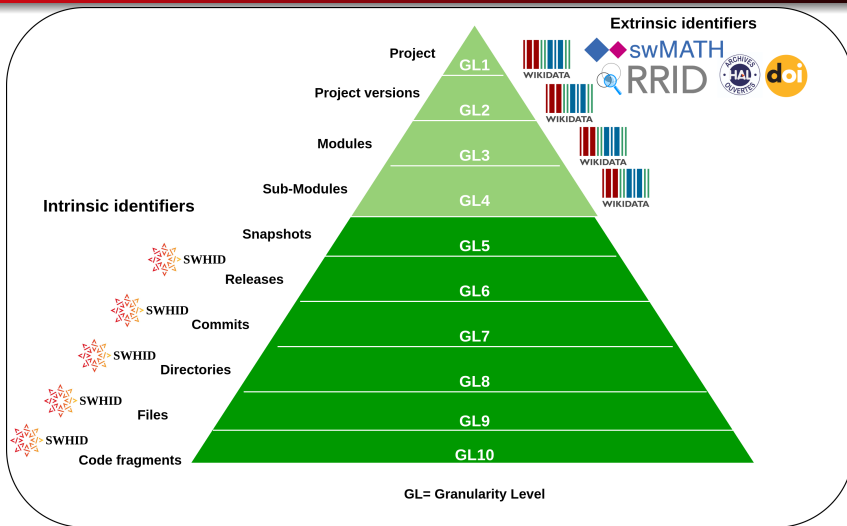
Top concept layers vs. bottom artifact layers

Under the hood: a uniform data model with intrinsic identifiers



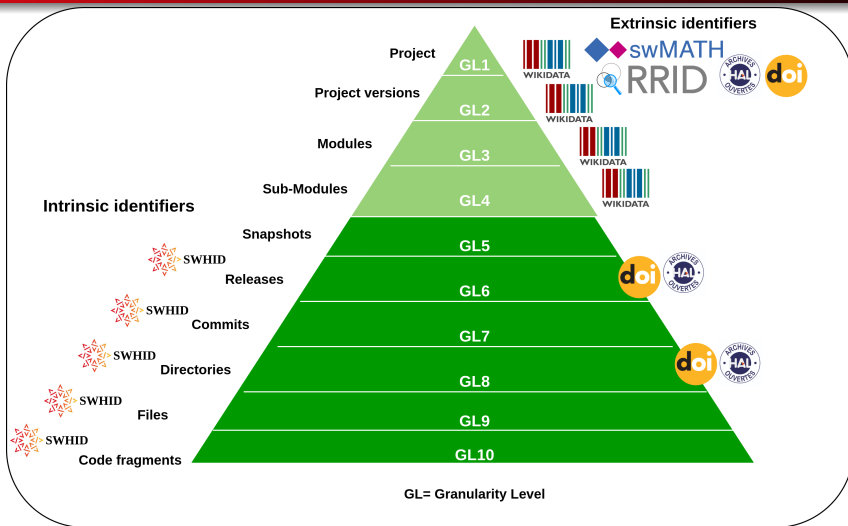
Top concept layers vs. bottom artifact layers

Under the hood: a uniform data model with intrinsic identifiers



Top concept layers vs. bottom artifact layers

Under the hood: a uniform data model with intrinsic identifiers



Top concept layers vs. bottom artifact layers

- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd**
- 12 Actions

MINISTÈRE
DE L'ENSEIGNEMENT
SUPÉRIEUR,
DE LA RECHERCHE
ET DE L'INNOVATION
*Liberté
Égalité
Fraternité*



National Roadmap of Research Infrastructures (2022-...)

Framework

"The strategic, scientific and financial challenges related to research infrastructures are such that sooner or later we will need to jointly program European and national investments in major research facilities."

Frédérique Vidal, Minister of Research, 2018

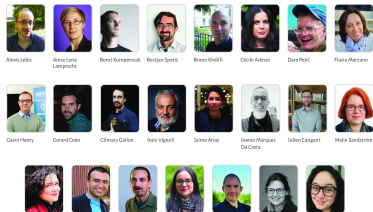
Software Heritage is now listed

A growing and active community

Team



Ambassadors



Contributors to the platform

```
[13:21:32] <seir> from last time i ran it? it very likely is
[13:21:47] <seir> we had a x2 on the edges in a single year
[13:23:14]
[13:53:44]
[13:54:01]
[13:55:11]
[13:55:23]
[13:55:31]
[13:55:44]
[14:02:57]
[15:19:35]
[15:19:59]
[15:20:48]
[15:20:59]
[15:21:18]
[15:21:31]
[15:23:25]
[15:23:40]
[15:23:44]
[15:24:13]
[15:25:00]

<seir> from last time i ran it? it very likely is
<seir> we had a x2 on the edges in a single year
<vorentz> ah
<zack> seir: i think i was remembering the LLP time on granet rather than the one (on the previous graph) on the big mem telecom machine
<zack> wasn't it something like 10-14 days (on granet)?
<seir> zack: it depends on the number of weights you use
<seir> i had something like that to do the parameter sweep
<seir> but then i settled on a few good gamma values
<seir> and afterwards it was only ever ~3-4 days
<zack> ok
<jelmer> vorentz: when is jenkins meant to kick in ? i didn't think the CI would mean you pasting test results in comments :P
<jelmer> alternatively, i could try to get it working locally - for some reason tox doesn't run here, complaining it can't find sw.h.scheduler
<vorentz> jenkins is down until tomorrow evening (paris time)
<vorentz> bad day for submitting your code :D
<vorentz> er yeah i just fixed that issue
<vorentz> but the fixed sw.h.scheduler is not pushed to pypi because jenkins
<jelmer> ah
<vorentz> in the meantime, you can change apply this patch: https://gitlab.softwareheritage.org/-/snippets/1546
<vorentz> as an ugly workaround
<vorentz> actually, just adding "pytest-postgresql < 4.0.0" should do it
<vorentz> when jenkins is back online i'll push a new release of sw.h.scheduler without the missing dependency on pytest-postgresql

Nicks
  # Alphare
  # ames
  # amj
  # amoeba
  # ar-jan
  # bchaudet[m]
  # cmatrix[m]
  # dan-
  # deesix
  # ericon2314
  # franckbret1
  # guerby
  # Guest92
  # haltcode2
  # hplar
  # jayeshiv
  # jelmer
  # KShivendu
  # landre[m]
  # marmoute
```

Awards



Antoine Pietri, best French PhD in Software Engineering "Enabling Big Code analysis on a very large source code corpus". Awarded by the CNRS research working group GPL <https://theses.hal.science/tel-03515795v1>



Stefano Zacchiroli with **Davide Rossi**, Google Award for Inclusion Research 2022, for the research project "What Causes the Lack of Diversity in Open Source?". <https://research.google/outreach/air-program/recipients/>



Antoine Pietri with **Stefano Zacchiroli**, Award Best Dataset Paper: "A Large-scale Dataset of (Open Source) License Text Variants". <https://arxiv.org/abs/2204.00256>

Annual report



Collecting, preserving
and sharing software
source code since 2015

5 years in 5 minutes

[Link](#)



Evolution of our codebase

[Link](#)



- 1 Introduction
- 2 Software and Open Science
- 3 (Bad) SOTA
- 4 Zoom on software specific aspects
- 5 An emerging policy framework
- 6 Assessing the needs and a strategy to address them
- 7 Meet Software Heritage: universal archive of source code
- 8 Demo time!
- 9 Practical tools for researchers
- 10 Shedding some light on the reference/identifiers confusion
- 11 Software Heritage, cont'd
- 12 Actions**

What this means for a PhD student/young researcher

Starting with your next paper

- **archive** the exact code behind every result (Save Code Now / deposit / webhook)
- **reference** claims at the right granularity: a SWHID for the revision, **directory**, the file, the lines
- **describe** it: a `codemeta.json` (and a generated `CITATION.cff`) in every repository
- **cite and credit** it in the bibliography, like any other research output, use and promote biblatex-software

As a reviewer or AEC member

- ask authors for *archival proof* (a SWHID), not a fragile URL
- one habit, applied every time, changes the norms of your community

The double-blind problem

Double-blind review needs your code — *without* revealing who wrote it.

`git-anonymize` : git-only, error-prone, leaks commit logs, makes a copy

A simpler, intrinsic answer

Share a *directory* SWHID of your code tree: `swh:1:dir:fdf3...a91c`

- **no identity**: hash *content alone* — no author, no origin, nothing to scrub
- **any technology**: a plain directory or tarball, not just git (Hg, SVN, no-VCS: all work)
- **zero overhead**: one command; reviewers can verify the bytes are exactly yours

```
swh identify --type directory ./ →  
swh:1:dir:fdf3...a91c
```

don't treat your code like an afterthought: *record, share and value it*
the tools are ready — starting with your next paper

Questions?

References



UNESCO, *Draft recommendations on Open Science*
2021, ([online](#))



French Ministry of Research, *Second National Plan for Open Science*
2021, ([online](#))



EOSC SIRS Task Force, *Scholarly Infrastructures for Research Software*
2020, Publications office of the European Commission, ([10.2777/28598](#))



R. Di Cosmo, *Archiving and Referencing Source Code with Software Heritage*
International Conference on Mathematical Software 2020 ([10.1007/978-3-030-52200-1_36](#))



J.F. Abramatic, R. Di Cosmo, S. Zacchiroli, *Building the Universal Archive of Source Code*
CACM, October 2018 ([10.1145/3183558](#))

Appendix

- 13 FAIR
- 14 HAL and SWH
- 15 Broader policy issues
- 16 Detailed addressing ARDC
- 17 Revolutionary infrastructure, scientific challenges
- 18 All the source code
- 19 Milestones, policy
- 20 Public code, mirrors

What about FAIR? (Findable, Accessible, Interoperable, Reusable)

FAIR data principles *for data*

in a nutshell: metadata, metadata, metadata all over the place (makes sense for data)

But software is *not data* ...

the terms *interoperability* and *reusability* have precise technical meaning for software, and *differ significantly* from what is intended by the I and R of FAIR;

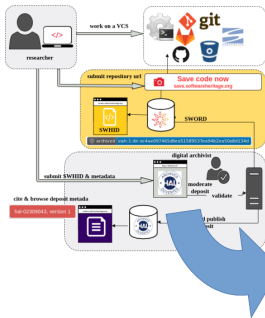
- see the entries for [software interoperability](#) and [software reusability](#)
- it is *very difficult* to achieve these properties even for commercial software developed by multi billion dollars corporations

FAIR for software is a distraction

let's focus on the real issues at stake: ARDC a good starting point

- 13 FAIR
- 14 HAL and SWH
- 15 Broader policy issues
- 16 Detailed addressing ARDC
- 17 Revolutionary infrastructure, scientific challenges
- 18 All the source code
- 19 Milestones, policy
- 20 Public code, mirrors

Overview of the Software Heritage / HAL synergy



<https://hal.archives-ouvertes.fr/hal-02130801>

[illegible]

Enter a SWHID to resolve or keyword(s) to search for it

<https://hal.archives-ouvertes.fr/hal-02130801>

14 June 2019, 13:43 UTC

< Code Branches () Releases (0) Vists

Revision: e8e18328952266b7875c692963b11963b1496107 393bd11 / linbox-1.6.3 / linbox / config-blas.h Raw File

Tip revision: e8e18328952266b7875c692963b11963b1496107 authored by Software Heritage on 11 June 2019, 08:12 UTC
hal: Deposit 297 in collection hal

config-blas.h

```

1 /* config-blas.h
2 * Copyright (C) 2005 Pascal Giorgi
3 *      2007 Clement Pernet
4 * Written by Pascal Giorgi <pgiorgi@waterloo.ca>
5 *
6 * =====LICENCE=====
7 * This file is part of the library LinBox.
8 *
9 * LinBox is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public
11 * License as published by the Free Software Foundation; either
12 * version 2.1 of the License, or (at your option) any later version.
13 *
14 * This library is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
17 * Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public
20 * License along with this library; if not, write to the Free Software
21 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
22 * =====LICENCE=====

```

#endif LINBOX config blas h

[swh:1:dir:393b611a1424f032e83569bf6762502371cfcf65](#)

Growing adoption of SWH in Academia (selection)

HAL software curated deposit workflow

Curated Archiving of Research Software Artifacts

International Journal of Digital Curation, 2020

Reference archive for swmath.org



See *code* links, e.g.

SemiPar package

IPOL (image processing)



- archive (deposit)
- reference
- BibLaTeX

eLife (life sciences)



- archive (save code now)
- reference

JTCAM (mechanics)

- [instructions for authors](#)
- biblatex-software in journal L^AT_EX class

Policy: France



*National Plan
for Open Science
and Research
Infrastructures*

Policy: Europe



EOSC SIRS report

- SWHIDs
- archive

Guidelines



Software Heritage

- 1 Prepare your public repository (README, AUTHORS & LICENSE files)
- 2 Save your code (<http://save.softwareheritage.org/>)
- 3 Reference your work (Full repository, specific version or code fragments)

- [summary](#)
- [ICMS 2020](#)

Call to action on ARDC: let's foster adoption!

Train students and colleagues to [archive and reference relevant source code](#)

- full details in the [ICMS 2020](#) article
- short operational [HOWTO online](#)

Engage conferences, journals, learned societies to use Software Heritage and SWHIDs

APIs for [save code now](#) and [deposit](#) are available to integrate with

- Research Articles
- Artifact Evaluation Committees
- Badging initiatives

Help grow and structure the community

- Promote the [ambassador program](#)
- Encourage our institutions to
 - include Software Heritage in their Open Science policy
 - become [member/sponsor](#)
 - build a Software Heritage mirror (see ENEA)

- 13 FAIR
- 14 HAL and SWH
- 15 **Broader policy issues**
- 16 Detailed addressing ARDC
- 17 Revolutionary infrastructure, scientific challenges
- 18 All the source code
- 19 Milestones, policy
- 20 Public code, mirrors



Ref.: CL/4363

Subject: Draft text of the UNESCO Recommendation on Open Science

Madam/Sir,

At its 40th session in November 2019, the UNESCO General Conference decided to elaborate a draft Recommendation on Open Science.

This was a major decision, which has since mobilized the entire Organization and all of its Member States in the development of this new standard-setting instrument.

After two years of joint work, this process is now entering its final phase, following the consensus reached on the draft text during the intergovernmental meeting of experts held from 6 to 11 May 2021.

I have the pleasure to submit to you this draft recommendation, which will be put forward for adoption at our next General Conference in November 2021.

The definitions and principles that it contains constitute a common – and currently unprecedented – framework to support scientific cooperation and make science more transparent, more accessible, more equitable and more inclusive.

For any further information, Sharmila Nair-Bedouelle, Assistant Director-General for Natural Sciences, is at your disposal at the following email address: openscience@unesco.org.

Thanking you for your commitment, please accept, Madam/Sir, the assurances of my highest consideration.


Audrey Azoulay
Director-General

Enclosure: 1

1. Draft text of the UNESCO Recommendation on Open Science

cc: Permanent Delegations to UNESCO
National Commissions for UNESCO

7, place de Fontenay
92628 Paris 15, France
Tel.: +33 (0)1 55 36 36 32
www.unesco.org

To Ministers responsible for relations with UNESCO

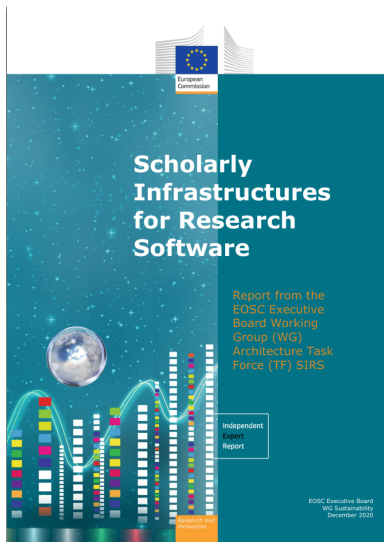
Selection from the recommendations

• Open Source for Open Science

"The source code must be included in the software release and made available on openly accessible repositories and the chosen license must allow modifications, derivative works and sharing under equal or compatible open terms and conditions"

• Infrastructures

"Open science infrastructures should be organized and financed upon an essentially not-for-profit and long-term vision, which enhance open science practices and guarantee permanent and unrestricted access to all, to the largest extent possible."

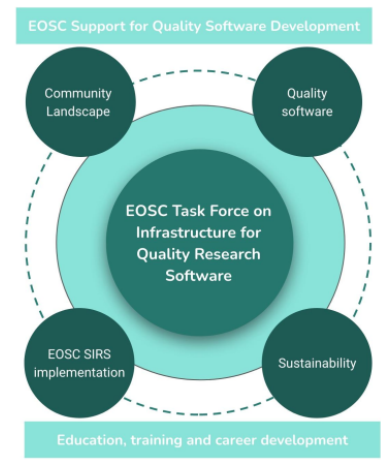


Important *policy tool* in Open Science (Dec 2020)

- 9 infrastructures
 - 3 archives
 - 3 open access publishers
 - 3 aggregators
- recommendations
 - archive in Software Heritage, use SWHID
 - open non profit
 - default to open source for research software

"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"

See <https://doi.org/10.2777/28598>



Ongoing action in the EOSC

Task force on infrastructures for quality research software

- Foster the development and deployment of tools and services that allow researchers to properly archive, reference, describe with proper metadata, share and reuse research software.
- Improve the quality of research software, both from the technical and organizational point of view ...
- Increase recognition to software developers and maintainers of research software ...

See [the charter of the task force](#).

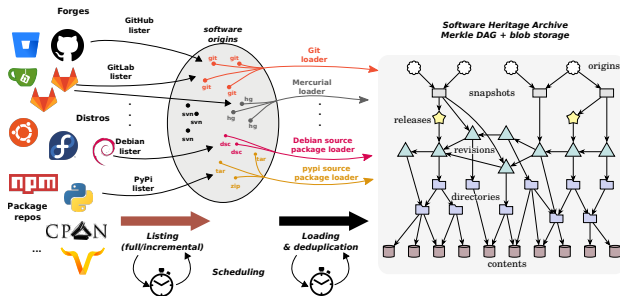
- 13 FAIR
- 14 HAL and SWH
- 15 Broader policy issues
- 16 Detailed addressing ARDC**
- 17 Revolutionary infrastructure, scientific challenges
- 18 All the source code
- 19 Milestones, policy
- 20 Public code, mirrors

Addressing the A(rchive)

Universal source code archive

not only research

(16B+ files, 260M+ projects)



- your research software *is likely there already!*
- anyone can trigger archival with save.softwareheritage.org
- selected partners can push to the archive via deposit.softwareheritage.org

Saving 250.000 endangered repositories...

- summer 2019: BitBucket announce Mercurial VCS phase out
- fall 2019: Software Heritage teams up with Octobus (funded by NLNet, thanks!)
- july 2020: BitBucket erases 250.000 repositories
- august 2020: bitbucket-archive.softwareheritage.org is live

... preserving the web of knowledge

(original tweet is [here](#))



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

Bottomline

explicit deposit is important, ...

... and we must promote it...

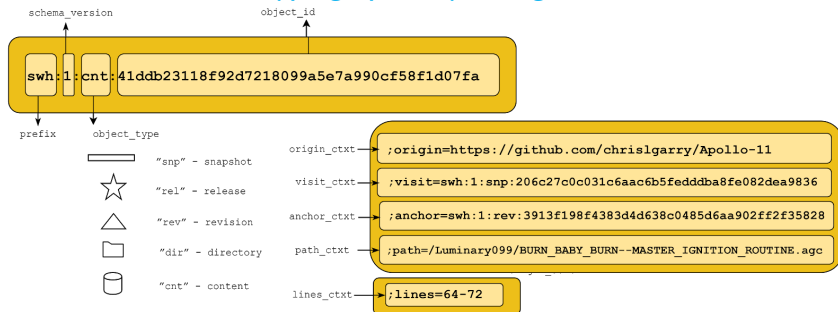
... but will never be enough.

(think also of all software dependencies!)

Software Hash Identifiers (SWHID)

[see swhid.org](https://swhid.org)

50+B intrinsic, decentralised, cryptographically strong identifiers, SWHIDs

In [SPDX 2.2](#); IANA "swh:"; WikiData [P6138](#); standard [ISO 18670](#)Full fledged *source code references* for traceability, integrity and reproducibilityExamples: [Apollo 11 AGC](#), [Quake III rsqrt](#); Guidelines available: [HOWTO](#) and [ICMS 2020](#)

Addressing D(escribe) and C(ite) in ARDC (see [ICMS 2020](#) for details)

Describe

- Collect *intrinsic metadata*
- Contributed the [Codemeta generator](#)

CodeMeta generator

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

The software itself

Name

the software title

Description

Creation date

First release date

Cite/Credit

- Contributed *software citation style*
[biblatex-software](#), v 1.2-2 now on [CTAN](#)



- 13 FAIR
- 14 HAL and SWH
- 15 Broader policy issues
- 16 Detailed addressing ARDC
- 17 Revolutionary infrastructure, scientific challenges**
- 18 All the source code
- 19 Milestones, policy
- 20 Public code, mirrors

Summing up: a revolutionary infrastructure *designed for source code*



Software Heritage

www.softwareheritage.org

global source code archive

Library of Alexandria of source code



- harvest *all* software source code
- on demand harvesting and *curated deposit*

universal intrinsic identifiers

SWHID standard is independent of version control systems

uniform data model, full graph of development history

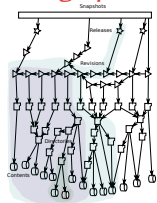
enables large scale, big code research

infrastructure for Open Science

base layer for software source code in the Open Science architecture

A revolutionary infrastructure *designed for software source code*

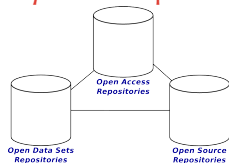
The *graph* of public software development



All software development
in a **single graph** ...

- enable traceability

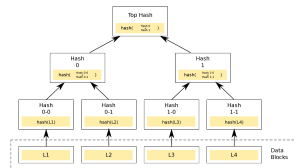
A *pillar* of Open Science



Reference **archive** of
Research Software

- reproducibility
- reference

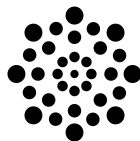
The *global ledger* of public code



... a **Merkle** graph

- ensure integrity

Reference platform for *Big Code* **uniform** data structure



- large scale studies
- cybersecurity
- machine learning, AI, ...

A challenging scientific and technical undertaking

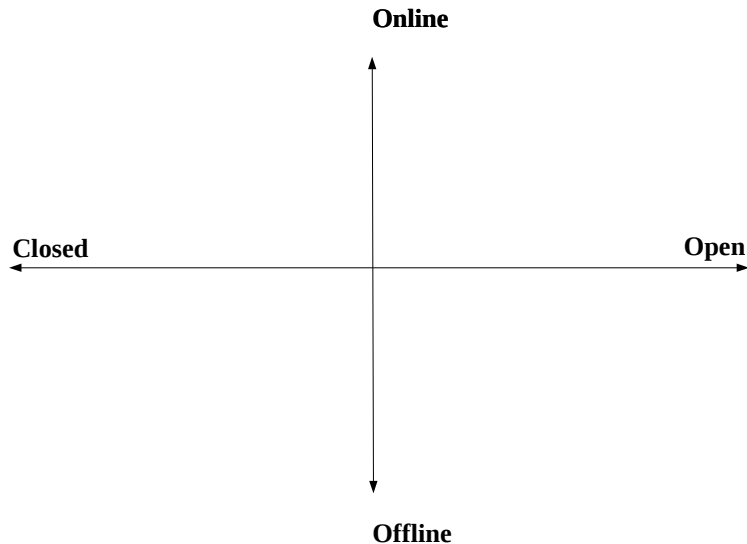
A novel, large infrastructure

- object storage with peculiar workload
- gigantic Merkle graph
- counting tens of billions of objects (reuse P. Flajolet's seminal work)
- and much more: see the 2021 technical roadmap

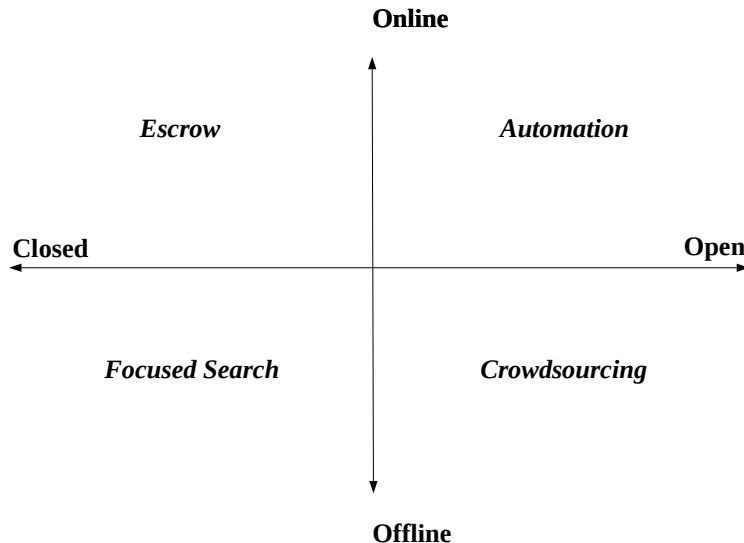
First datasets are available for Big Code analysis

- full graph of software development (~20Bn nodes, ~200Bn edges) see Pietri, Spinellis, Zacchiroli, MSR 2019
<https://dx.doi.org/10.1109/MSR.2019.00030>
- MSR 2020 mining competition see <https://2020.msrconf.org/track/msr-2020-mining-challenge#Call-for-Papers>

- 13 FAIR
- 14 HAL and SWH
- 15 Broader policy issues
- 16 Detailed addressing ARDC
- 17 Revolutionary infrastructure, scientific challenges
- 18 All the source code**
- 19 Milestones, policy
- 20 Public code, mirrors



All the source: strategies



- 13 FAIR
- 14 HAL and SWH
- 15 Broader policy issues
- 16 Detailed addressing ARDC
- 17 Revolutionary infrastructure, scientific challenges
- 18 All the source code
- 19 Milestones, policy**
- 20 Public code, mirrors

Summer 2015



The collection starts: first server, (very) early prototype

June 30th 2016



Public unveiling, with the first sponsors: Microsoft and DANS

April 3rd 2017



Unesco - Inria agreement on software access and preservation.

June 7th 2018



Opening the archive to the world

February 26th 2019



Publication of the expert meeting
**Paris Call on
Software Source Code**

June 2019



Software Heritage Graph as **Open Dataset on AWS**

"Upload filters": a threat to *all modern software development*

- developing platforms (GitHub, GitLab, Bitbucket, etc.)
- distribution platforms (Maven, Pypi, CRAN, CTAN, etc.)
- archives (Software Heritage)

We got an exclusion for

~~non for profit~~ open source software developing *and sharing* platforms

Key role of Software Heritage

policy-maker awareness, essential insights for NGOs, government contacts

- 13 FAIR
- 14 HAL and SWH
- 15 Broader policy issues
- 16 Detailed addressing ARDC
- 17 Revolutionary infrastructure, scientific challenges
- 18 All the source code
- 19 Milestones, policy
- 20 Public code, mirrors



code.etalab.gouv.fr (alpha)

Contact

Glossary

About

Etalab

Public sector source codes

This website lets you browse some of the source codes opened by public bodies. If your source code is not referenced on this website, [send us](#) a link to your repository.

Source code repositories

Organizations or groups

Figures

Free search

License

Language

☐ Forks only

☐ Hide archives

☐ With a description

☐ With known license

3669 repositories

Repository / group	Archive	Description	Updated	Forks	Stars	Issues
medle / SocialGouv		MedLé : plateforme permettant aux établissements de santé de déclarer leur activité médico-légale	11/8/2019	0	0	
reseauchaleur / deal-datalab			11/8/2019	0	0	

<https://code.etalab.gouv.fr>

Thomas Jefferson, February 18, 1791

...let us save what remains: not by vaults and locks which fence them from the public eye and use in consigning them to the waste of time, but by such a multiplication of copies, as shall place them beyond the reach of accident.

Welcoming ENEA

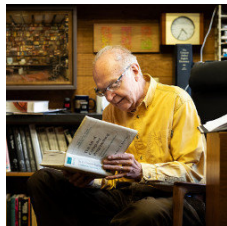


Italian National Agency for New Technologies,
Energy and Sustainable Economic Development

- first **institutional** mirror
- increased resilience
- **AI infrastructure** for researchers
- stepping stone to
an European joint effort

Calling for preservation: Donald Knuth and Len Shustek

Communications of the ACM, February 2021



"Telling historical stories is the best way to teach. It's much easier to understand something if you know the threads it is connected to."

Let's Not Dumb Down the History of Computer Science

Donald E. Knuth, Len Shustek

<https://doi.org/10.1145/3442377>

A unique opportunity

most of the creators are still here: we can talk to them!

but the clock is ticking...

Source code history for Security and Transparency

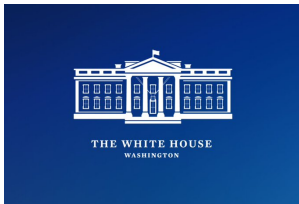
Where does reused software come from?



Do *you* know where it comes from?

- the software you ship
- the software you use
- the software you acquire
- the software that
 - has that bug
 - has that vulnerability

KYSW: Know Your SoftWare



Like KYC in banking, KYSW is now essential all over IT...

Sec. 4. Enhancing Software Supply Chain Security

ensuring and attesting, to the extent practicable, to the integrity and provenance of open source software

May 2021 POTUS Executive Order