Logiciels et Codes sources

pour la Science Ouverte

Roberto Di Cosmo

Director, Software Heritage
Inria and Université de Paris

8 Décembre 2021
CoSO, Collège Données

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Outline

1. Introduction
2. Software Source Code and Open Science
3. Adressing the needs
4. Demo time!
5. The way forward

Short Bio: Roberto Di Cosmo

Computer Science professor in Paris, now working at INRIA

- *30 years* of research (Theor. CS, Programming, Software Engineering, Erdos #: 3)
- *20 years* of Free and Open Source Software
- *10 years* building and directing structures for the common good

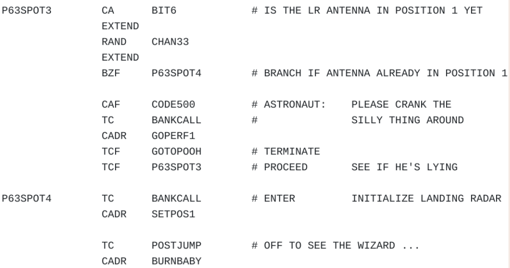
1999	*DemoLinux* – first live GNU/Linux distro
2007	*Free Software Thematic Group* 150 members 40 projects 200Me
2008	*Mancoosi project* www.mancoosi.org
2010	*IRILL* www.irill.org
2015	*Software Heritage* at INRIA
2018	*National Committee for Open Science*, France

Why Software *Source Code* matters

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)



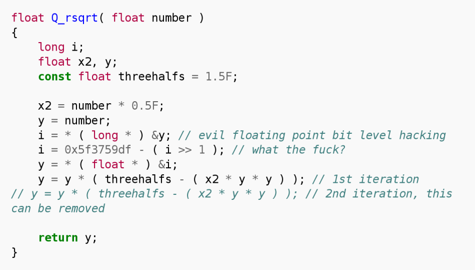
```
P63SPOT3        CA      BIT6            # IS THE LR ANTENNA IN POSITION 1 YET
                EXTEND
                RAND    CHAN33
                EXTEND
                BZF     P63SPOT4        # BRANCH IF ANTENNA ALREADY IN POSITION 1

                CAF     CODE500         # ASTRONAUT:     PLEASE CRANK THE
                TC      BANKCALL        #                SILLY THING AROUND
                CADR    GOPERF1
                TCF     GOTOPOOH        # TERMINATE
                TCF     P63SPOT3        # PROCEED        SEE IF HE'S LYING

P63SPOT4        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.”

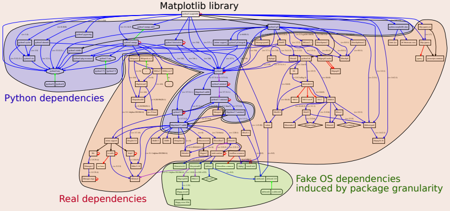
Source code is *special* (software is *not* data)

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*



Legal status

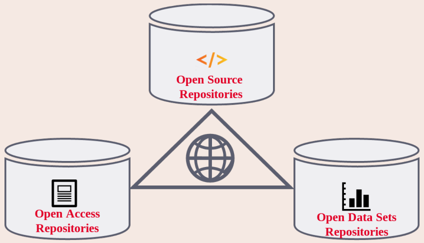
- software is covered by *copyright*, like articles, and unlike data
- there are special provision for software too (it is not *exactly* like articles or books!)

Outline

1. Introduction
2. Software Source Code and Open Science
3. Adressing the needs
4. Demo time!
5. The way forward

Software Source code: pillar of Open Science

Three pillars of Open Science



Open Source Repositories

Open Access Repositories

Open Data Sets Repositories

A plurality of needs

Researcher

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

Laboratory/team track software contributions

- produce reports / web page

Research Organization know its software assets

- technology transfer
- impact metrics

What is at stake: ARDC in increasing order of difficulty

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*!)

We need an infrastructure *designed for* software source code: *now we have one!*

What is at stake: beyond ARDC

Sustainability, technology transfer

Organisational schemas, legal tools, economic models, processes and policies to ensure research software can be maintained and sustained over time, maybe in connection with industry

Evaluation (funding, careers, etc.)

- avoid the numbers game (beware of *naive software citation counting*)
- identify *roles* in software projects, see:

  P. Alliez, R. Di Cosmo, B. Guedj, A. Girault, M.-S. Hacid, A. Legrand and N. Rougier
  Attributing and referencing (research) software: Best practices and outlook from Inria,
  CiSE 2020 (10.1109/MCSE.2019.2949413)

Regulations are coming

software management plans, licensing recommendations, metadata and identification standards

Outline

1. Introduction
2. Software Source Code and Open Science
3. Adressing the needs
4. Demo time!
5. The way forward

Some key notions

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!

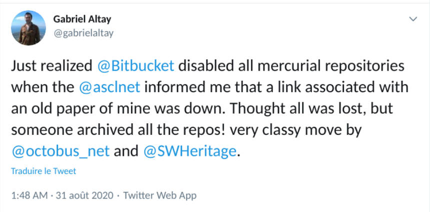
2015: the bad news

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

Summer 2019: BitBucket announce Mercurial VCS sunset

- 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 (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!)

Traditional archives are not adapted to software

Usual archival approach. . .

- independent information package(s)
- (persistent) identifier with a registry
- metadata record

. . . not well adapted to software source code . . .

- broad dependencies on non academic software
- full development history:
  - not just releases
- software development moved to *intrinsic identifiers* (more on this later)
  - putting a DOI on a .zip file does not fit the bill

. . . we can do better

use Software Heritage: it is *designed for source code*

Outline

1. Introduction
2. Software Source Code and Open Science
3. Adressing the needs
4. Demo time!
5. The way forward

Focus on Archive and Reference

- Browse the archive
- Trigger archival of your preferred software in a breeze
- Get and use SWHIDs (full specification available online)
- Deposit via HAL, e.g.
  - LinBox, SLALOM, Givaro, NS2DDV, SumGra, Coq proof, . . .
- Example use in a research article: compare Fig. 1 and conclusions
  - in the 2012 version
  - in the updated version using SWHIDs and Software Heritage
- Cite software with the biblatex-software style from CTAN
- Example use in a research article: extensive use of SWHIDs in a replication experiment
- Example in a real journal: an article from IPOL
- Supporting reproducible builds: Guix and Nix

Growing adoption of SWH in Academia (selection)

HAL software curated deposit workflow

Curated Archiving of Research Software Artifacts
International Journal of Digical Curation, 2020

Reference archive for swmath.org

swMATH
an information service for mathematical software

See *code* links, e.g. SemiPar package

IPOL (image processing)

- archive (deposit)
- reference
- BibLaTeX

eLife (life sciences)

- archive (save code now)
- reference

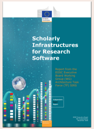
JTCAM (mechanics)

- instructions for authors
- biblatex-software in journal LaTeX class

Policy: France

National Plan for Open Science

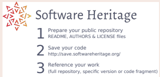
Policy: Europe



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 fragment)

- summary
- ICMS 2020

Outline

1. Introduction
2. Software Source Code and Open Science
3. Adressing the needs
4. Demo time!
5. The way forward

Le Collège Logiciel dans le CoSO

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

Second French Plan
for Open Science

GENERALISING
OPEN SCIENCE
IN FRANCE 2021-2024

2nd National Plan for Open Science (6/7/2021)

Open and promote research software source code

- actions (selection)
  - charter for research software policy
  - recognize software development (see announcement of the 2021 prize)
  - coordinate communities of practice
  - build a connected ecosystem of research outputs
- recommendations (selection)
  - archive in Software Heritage
  - standardise and use SWHID
  - build a national catalog of research software
  - leverage ADAC network

See official announcement

Two pronged approach: 1, Process and Expertise

Develop a strategy to address these issues

- build a corpus of shared knowledge
- build a network of expertise
  - connect with open source experts
  - connect with other institutions
  - connect with OSPOS
- make informed strategic decisions
- develop a decision tree for researchers

How to proceed

- join the upcoming CoSO software group
- connect with international organizations

Two pronged approach: 2, Describe and Track

Build a *uniform, global catalog* of research software

- standard metadata to encode all the relevant information
- single entry point and process to enter and extract information
- contains information on all research software, open or closed
- some information may not be public (e.g. tech transfer details)

What we have

- HAL and SWH: curated deposit *for open code* with *public metadata*
- contributor roles: from Inria and INS2I
- pushed to international level (via EOSC, RDA, Force11)

What we need

- *massive import* of existing information on open code
- expand catalog to cover *closed code and private information*
- collaboration with tech transfer teams

We can build toghether what is missing, in a joint project

Questions?

References

- Software Heritage, *"Five years in five minutes"*
  2021, (official video for the 5 year anniversary at UNESCO)

- MESRI, *Plan National pour la Science Ouverte*
  2021, (official announcement)

- EOSC SIRS Task Force, *Scholarly Infrastructures for Research Software*
  2020, European Commission, (10.2777/28598)

- R. Di Cosmo, *Archiving and Referencing Source Code with Software Heritage*
  ICMS 2020 (10.1007/978-3-030-52200-1_36). See also the HOWTO for researchers online.

- R. Di Cosmo, M. Gruenpeter, S. Zacchiroli
  Referencing Source Code Artifacts: a Separate Concern in Software Citation,
  CiSE 2020 (10.1109/MCSE.2019.2963148) (hal-02446202)

- P. Alliez, R. Di Cosmo, B. Guedj, A. Girault, M.-S. Hacid, A. Legrand and N. Rougier
  Attributing and referencing (research) software: Best practices and outlook from Inria,
  CiSE 2020 (10.1109/MCSE.2019.2949413) (hal-02135891)

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

Appendix

Outline

6 Source code, and FAIR

7 Phase 1: focus on ARDC

8 Phase 2: ARDC and beyond

Software and Software *Source Code*

> *“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
4004fb:  c3
```

Program (source code)

```
/* Hello World program */

#include<stdio.h>

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

Software is *special*, cont'd

Software as a concept

- software project / entity
- the creators and the community around the project
- the software solution / functionality

Software artifact

- the binaries for different environments
- the software source code for each version
  - the multiple files or code fragments

Versioning, granularity

Project "Inria created OCaml and Scikit-learn"

Release "2D Voronoi Diagrams were introduced in CGAL 3.1.0"

Precise state of a project "This result was produced using commit 0064fbd…"

Code fragment "The core algorithm is in lines 101 to 143 of the file parmap.ml contained in the precise state of the project corresponding to commit 0064fbd…"

What about FAIR?

FAIR data principles *for data*

in a nutshell: metadata, metadata, metadata all over the place to make sense of data

But software is *not data* . . .

- a source code repository usually contains significant metadata by itself
- 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 multinationals

Bottomline

- "making software FAIR" is not the key issue at stake
- need to focus on more actionable properties: ARDC is a good starting point

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

Outline

6 Source code, and FAIR

7 Phase 1: focus on ARDC

8 Phase 2: ARDC and beyond

Software Heritage in a nutshell

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

Debian
Sourceforge
CPAN
Gitorious
Maven
Inria
Bitbucket
GitHub
BerliOs
CTAN
GitLab
CRAN
GoogleCode
Adullact

find and reference all software source code

Universal archive

damage
storage
disaster
reference
deletion
media
malicious
obsolete
aging
attack
tear
dependencies
dangling
wear
corruption
encryption
format

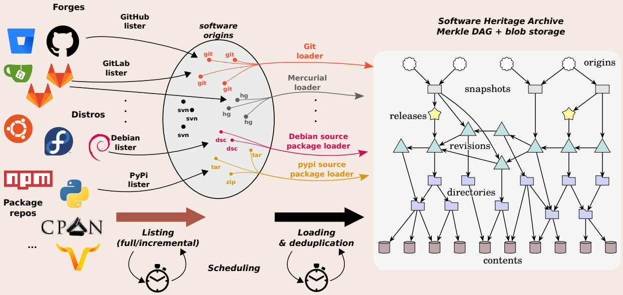
preserve all software source code

Research infrastructure

enable analysis of all software source code

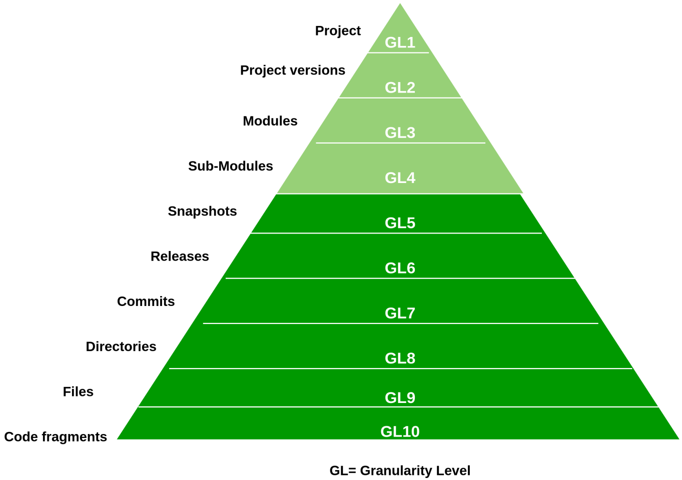
A(rchive)

Universal source code archive *not only research* (11B+ files, 160M+ 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

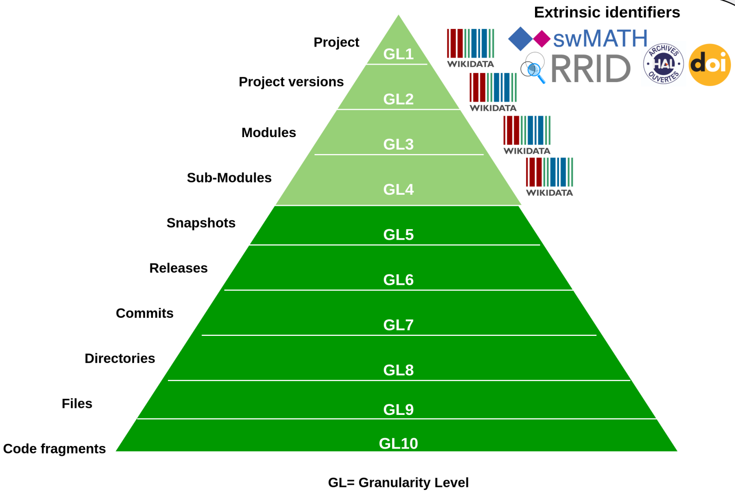
R(eference): granularity and identifiers 10.15497/RDA00053



Top concept layers vs. bottom artifact layers

R(eference): granularity and identifiers

10.15497/RDA00053



Extrinsic identifiers

swMATH

RRID

WIKIDATA

Project — GL1

Project versions — GL2

Modules — GL3

Sub-Modules — GL4

Snapshots — GL5

Releases — GL6

Commits — GL7

Directories — GL8

Files — GL9

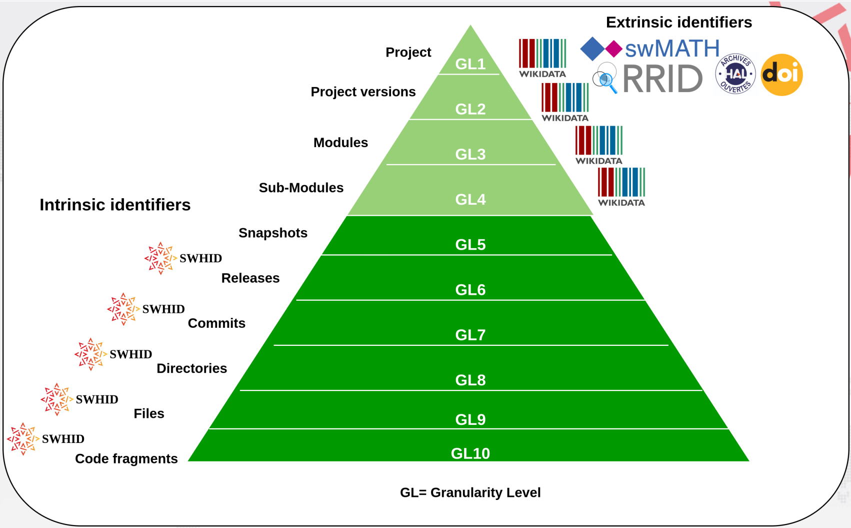
Code fragments — GL10

GL= Granularity Level

Top concept layers vs. bottom artifact layers

R(eference): granularity and identifiers

10.15497/RDA00053



Extrinsic identifiers

WIKIDATA swMATH RRID HAL doi

Project GL1

Project versions GL2

Modules GL3

Sub-Modules GL4

Intrinsic identifiers

Snapshots GL5

SWHID

Releases GL6

SWHID

Commits GL7

SWHID

Directories GL8

SWHID

Files GL9

SWHID

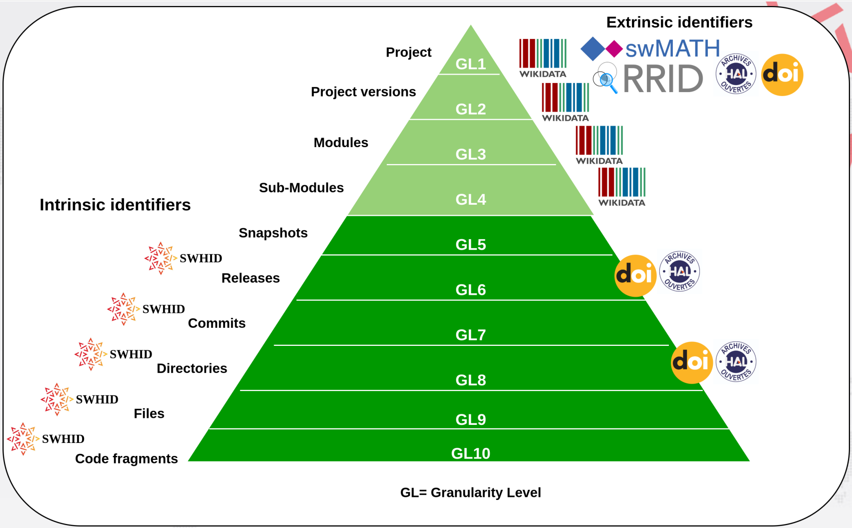
Code fragments GL10

GL= Granularity Level

Top concept layers vs. bottom artifact layers

R(eference): granularity and identifiers

10.15497/RDA00053



Top concept layers vs. bottom artifact layers

Extrinsic and Intrinsic identifiers in a nutshell

Extrinsic identifiers: no *per se* relation with the designated Object

A *register* keeps the correspondence between the identifier and the object

pre-internet era passport number, social security number, ISBN, ISSN, etc.

internet era DOI, Handle, Ark, PURLs, RRID, etc.

Intrinsic identifiers: derived from the designated Object

No register needed to keep the correspondence between the identifier and the object

pre-internet era musical notation, chemical notation (*NaCl* is table salt)

internet era cryptographic hashes for distributed software development, Bitcoin

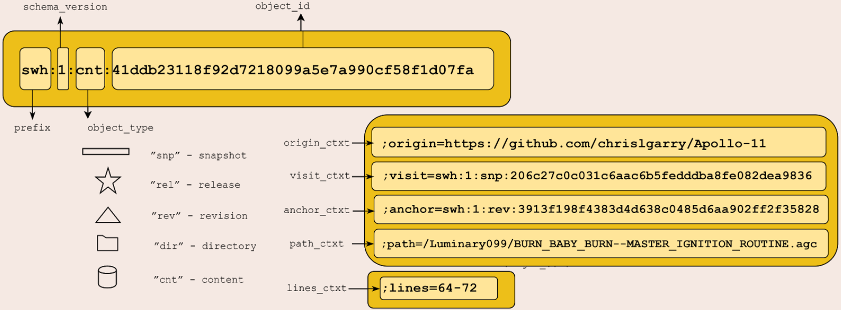
more in this dedicated blog post (with pointers to literature)

Meet the SWHID intrinsic identifiers

Software Heritage Identifiers (SWHID)

link to full docs

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



schema_version

object_id

swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa

prefix

object_type

"snp" - snapshot

"rel" - release

"rev" - revision

"dir" - directory

"cnt" - content

origin_ctxt ;origin=https://github.com/chrislgarry/Apollo-11

visit_ctxt ;visit=swh:1:snp:206c27c0c031c6aac6b5fedddba8fe082dea9836

anchor_ctxt ;anchor=swh:1:rev:3913f198f4383d4d638c0485d6aa902ff2f35828

path_ctxt ;path=/Luminary099/BURN_BABY_BURN--MASTER_IGNITION_ROUTINE.agc

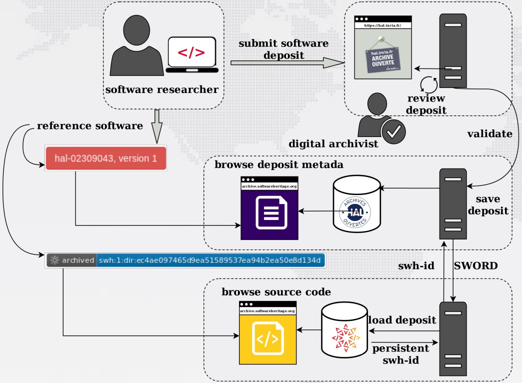
lines_ctxt ;lines=64-72

Emerging standard : Linux Foundation SPDX 2.2; IANA registered; WikiData P6138

Full fledged *source code references* for reproducibility

Examples: Apollo 11 AGC excerpt, Quake III rsqrt; Guidelines available, see ICMS 2020

D(escribe) and C(redit/cite): HAL+SWH



Deposit software in HAL poster

Generic mechanism:

- SWORD 2.0, review process, versioning

How to do it: (*guide*)

- deposit .zip or .tar.gz file with metadata
- new: deposit metadata on SWHID

Timeline:

- *Mars 2018*: test phase on HAL-Inria
- *September 2018*: open to all HAL
- *June 2021*:
  - 600+ source code deposits
  - metadata deposit on HAL-Inria
  - citation/metadata in BibTeX and CodeMeta

Outline

6 Source code, and FAIR

7 Phase 1: focus on ARDC

8 Phase 2: ARDC and beyond

Software management plans: there is more than meets the eye!

Sustainability

- economic model
- community and governance
- license

Evaluation and profit sharing

- make software count in careers and evaluations

Technical

- infrastructure, tools, processes, quality assurance

A license is not a business model, a forge is not a community

Cedric Thomas, OW2 CEO

Recall: beyond ARDC

Policy for dissemination and reuse

- open source research software
- revisit technology transfer and industry collaboration

Framework for evaluation and recognition

- make software development count in a career…
  - not the case in many countries (e.g. Italy)
- … but counting citations and commits *is not the silver bullet*

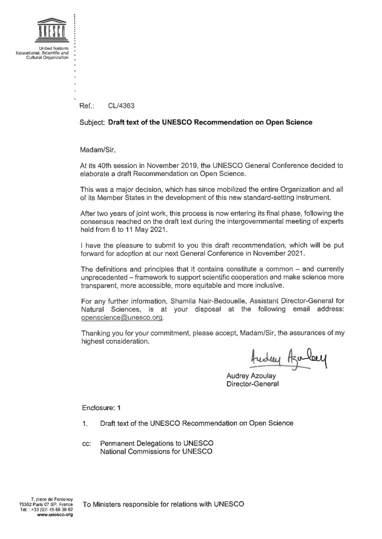
Sustainability

technical improve quality of *key* research software

financial make research software as easy to fund as buying a license (somewhat similar issues with Open Access)

Infrastructures, technologies and tools

The UNESCO recommendations for Open Science, 2018-2021



United Nations
Educational, Scientific and
Cultural Organization

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, Shamila 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

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 Fontenoy
75352 Paris 07 SP, France
Tél. : +33 (0)1 45 68 38 62
www.unesco.org

To Ministers responsible for relations with UNESCO

Selection from the recommendations

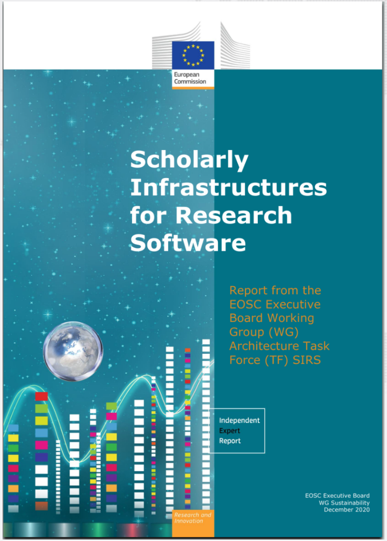
- Open Source for Open Science

  "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."

The EOSC SIRS report: Software Source Code and Open Science, 2020



European Commission

Scholarly Infrastructures for Research Software

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

Independent Expert Report

EOSC Executive Board
WG Sustainability
December 2020

Research and Innovation

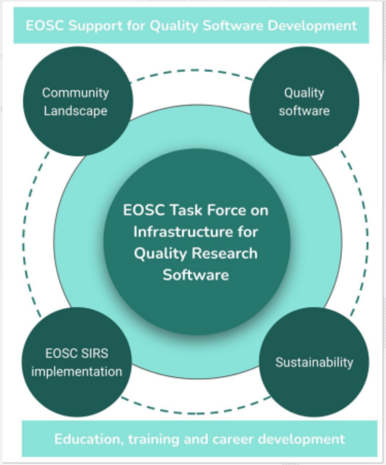
Important *policy tool* in Open Science (Dec 2020)

- 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

Software in the EOSC


EOSC Support for Quality Software Development
Community Landscape
Quality software
EOSC Task Force on Infrastructure for Quality Research Software
EOSC SIRS implementation
Sustainability
Education, training and career development

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.