Codes sources, logiciels libres

introduction et survol de quelques thémes clé

Roberto Di Cosmo
Inria and Université de Paris

Reseau des Administrateurs

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Outline

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

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");
}
```

Algorithm

Well-defined, step-by-step procedure to solve a problem

Problem: Find the position of a value x in an array A

Algorithm: Linear search

- start from the first position
- compare x with each element of A
  - if x matches, return the position
  - if no match is found, return failure

Remarks

- high level description, not immediately executable
- *existed well before computers were available*
  - strong connections with mathematics
  - Euclid's PGCD, Sieve of Eratosthenes, etc.

Pseudocode

Algorithm written as a simplified version of a program

```
procedure linearsearch (array, value)
  for each item in the array
   if item == value
     return the item's location
   end if
  end for
  return failure
end procedure
```

Remarks

- not yet detailed enough to be run on a machine
- typically found in algorithm textbooks
- useful to study the complexity of an algorithm

Program (source code)

Exact code written in a programming language, with all the details

```
// C++ code for linear search of x in arr[].  If x
// is present  then return its  location,  otherwise
// return -1
int search(int arr[], int n, int x)
{
 int i;
 for (i = 0; i < n; i++)
   if (arr[i] == x)
    return i;
 return -1;
}
```

Outline

Why Software *Source Code*

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."

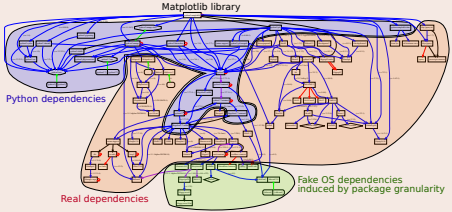
Source code is *special* (software is *not* data)

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*



Precious, endangered *executable* and *human readable* knowledge

key people passing away, platforms (GoogleCode, Gitorious, etc.) closing down …

no organised effort to catalog and archive it

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…"

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)

Bottomline

explicit deposit is important, …

… and we must promote it…

… but will never be enough.

(think also of all software dependencies!)

Outline

Course Notes

See https://www.dicosmo.org/CourseNotes/LogicielLibre/ for full course notes

En France

Le droit d'auteur est fondé sur les textes contenus dans le *Code de la Propriété Intellectuelle (CPI)*, qui est accessible sur `http://www.legifrance.gouv.fr`.

Un droit de propriété exclusif, et automatique

Art. L111-1:

> *L'auteur d'une oeuvre de l'esprit jouit sur cette oeuvre, du seul fait de sa création, d'un droit de propriété incorporelle exclusif et opposable à tous.*

Deux facettes: droit moral et droit patrimonial

droit moral (L121-1…L121-9)

L'auteur jouit du droit au respect de son nom, de sa qualité et de son oeuvre.
Ce droit est attaché à sa personne.
Il est perpétuel, inaliénable et imprescriptible. (Art. L121-1)

droit patrimonial (L122-1…L122-12)

Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite. (Art. L122-4)

Les logiciels relèvent de ce droit…

Art. L112-2:

Sont considérés notamment comme oeuvres de l'esprit au sens du présent code :

…

13 Les logiciels, y compris le matériel de conception préparatoire ;

…

Donc, il y a bien un droit moral et un droit patrimonial sur le logiciel, comme sur tout autre *oeuvre de l'esprit*.

Mais avec des restrictions aux droit patrimonial *pour les salariés*

Un logiciel est plus complexe qu'un livre, dans le temps et dans l'espace. Donc on introduit des exceptions au faveur des entreprises:

Art 113-9

Sauf dispositions statutaires ou stipulations contraires, les droits patrimoniaux sur les logiciels et leur documentation créés par un ou plusieurs employés dans l'exercice de leurs fonctions ou d'après les instructions de leur employeur sont dévolus à l'employeur qui est seul habilité à les exercer.

L'exception à l'exception: le cas des stagiaires

Pourtant, le logiciel dévéloppé par un non salarié, e.g. dans le cadre d'un stage *lui appartient*!

Cas des stagiaries

Si

- Le stage s'inscrit dans le cursus pour l'obtention d'un diplôme ou d'un titre.
- Le stagiaire reste sous la responsabilité juridique de l'établissement d'enseignement bien que hors de celui-ci.
- Il y a une convention de stage.
- Il n'y a pas de rémunération (la gratification n'est pas une rémunération)

Alors le stagiaire n'est pas considéré comme un salarié.

Voir: circulaire ministérielle du 30/10/1959, et celles n. 22 du 26/03/1970 et n. 86.065 du 13/02/1986.

Et des exceptions au droit moral, toujours *pour les salariés*

Art. L121-7

Sauf stipulation contraire plus favorable à l'auteur d'un logiciel, celui-ci ne peut :

- S'opposer à la modification du logiciel par le cessionnaire des droits mentionnés au 2° de l'article L. 122-6, lorsqu'elle n'est préjudiciable ni à son honneur ni à sa réputation ;
- Exercer son droit de repentir ou de retrait.

Le droit à la paternité est maintenu: on n'oublie pas de marquer son nom dans le code que l'on écrit!

En droit français ...

Ce qu'on appelle généralement *licence logicielle* est plus précisement un

Contrat de mise à disposition de logiciels

contrat accord privé entre des parties

mise à disposition il ne s'agit pas d'une "vente" ou "cession de droits", mais une simple "mise à disposition" dans des conditions précisées par le contrat lui même.

Ces conditions sont limitées seulement par l'accord des parties et par le droit national applicable.

Important: on peut avoir multiplicité de licences

Le detenteur des droits peut distribuer son logiciel sous une variété de licences différentes: la licence est un contrat de droit privé, et les ayant droits sont libres d'en signer autant qu'ils le souhaitent.

Très utilisé dans le monde du logiciel libre

Mozilla/Firefox code sous licence MPL/GPL/LGPL, au choix libre de l'utilisateur

OCaml compiler QPL pour tous, mais autre licence possible via le Consortium

Et aussi dans le logiciel propriétaire

- Windows peut être OEM ou pas

Outline

In a nutshell

Software that offers to *its users* the freedom to:

0. use the software
1. study and adapt the software
2. distribute software copies
3. distribute modified copies

Free Software has changed the way software is:

- developed
- tested
- deployed
- maintained
- marketed
- sold
- designed
- taught
- ...

Open Source vs. Free Software

Le paysage est assez varié:

libre :  Richard Stallman

copylefted : GPL/LGPL, etc.

non copylefted : BSD/MIT, etc.

open source : open source Bruce Perenes/Eric Raymond

Open Source Definition en 10 points.

Outline

Free Software Definition http://www.gnu.org/philosophy/

L'expression "Logiciel libre" fait référence à la liberté pour les utilisateurs d'exécuter, de copier, de distribuer, d'étudier, de modifier et d'améliorer le logiciel. Plus précisément, elle fait référence à quatre types de liberté pour l'utilisateur du logiciel :

- La liberté d'exécuter le programme, pour tous les usages (liberté 0).
- La liberté d'étudier le fonctionnement du programme, et de l'adapter à vos besoins (liberté 1). Pour ceci l'accès au code source est une condition requise.
- La liberté de redistribuer des copies, donc d'aider votre voisin, (liberté 2).
- La liberté d'améliorer le programme et de publier vos améliorations, pour en faire profiter toute la communauté (liberté 3). Pour ceci l'accès au code source est une condition requise.

Quelques remarques

- fort accent sur la *liberté* de l'utilisateur et du développeur
- pas d'antinomie avec le "commerce"

  > *"Logiciel libre" ne signifie pas "non commercial". Un logiciel libre doit être disponible pour un usage commercial, pour le développement commercial et la distribution commerciale.*

- les quatre libertés ne sont pas une "test suite" pour les licences (c'est voulu)

Pas de test suite pour la “liberté”

http://www.gnu.org/philosophy/free-sw.fr.html

Enfin, notez que les critères tels que ceux développés dans cette définition du logiciel libre demandent une réflexion sérieuse quant à leur interprétation. Pour décider si une licence de logiciel particulière est définie comme libre, nous la jugeons sur ces critères pour déterminer si elle convient à leur esprit tout comme à leur formulation précise. Si une licence inclut des restrictions inacceptables, nous la rejetons même si nous n’avons pas anticipé le problème dans ces critères.

Outline

The Open Source Definition

Origine

- dérivée des Debian Free Software Guidelines, rédigées en

1997 à partir d'une prémière version écrite par Bruce Perens (Debian, Busybox, Electric Fence).

- 10 critères précis (à approfondir dans un autre cours)

Organisation

- Open Source Initiative `www.opensource.org`, une "California public benefit corporation", née de l'initiative en 1998 de
  - Jon "maddog" Hall (Linux International)
  - Larry Augustin (VA Software)
  - Eric S. Raymond et Bruce Perens
- mission: maintenir la liste des licences qui respectent la définition.

En pratique

Différence d'approche ...

- Logiciel Libre: liberté de l'utilisateur
- Open Source: assurer le bon fonctionnement du dévéloppement collaboratif

... mais pas vraiment dans les licences reconnues

- presque aucun cas de différence

Outline

Licences

libre : *quelques dizaines de licences*, généralement très permissives, *pour des centaines de milliers de logiciels*…; on vous laisse modifier, distribuer, vendre et utiliser le logiciel, sous des conditions pas toujours équivalentes, d'où le *problème de compatibilité des licences*, et l'importance d'*éviter la prolifération des licences*;

propriétaire : *une licence par produit*, souvent très restrictive, on ne peut pas faire grand chose avec ce logiciel à part l'executer, et encore…

Le logiciel propriétaire n'est pas *plus simple* que le logiciel libre, il est *plus restrictif*.

Quelques caractéristiques des licences libres

Ayant une “oeuvre” (logiciel, bout de code) libre A,

- on peut *toujours* réaliser une “oeuvre dérivée” B (on change le code de A, ou on incorpore dans B du code de A), si la licence de A le permet;
- mais le statut de l”’oeuvre dérivée” change selon la licence de A

Impact sur les oeuvres dérivées

- A sous licence X ou BSD (version récente): presque aucune contrainte sur B
- A sous licence GPL: B doit être rediffusé sous licence GPL (copyleft)
- A sous licence LGPL: selon les cas, B peut être rediffusé sous une licence non LGPL, mais avec des contraintes techniques

La question des contributeurs et de la centralisation des droits

Cas typique

- une équipe de recherche diffuse son logiciel L sous licence GPL
- elle a du succès, elle accepte des contributions sans poser des questions (par exemple de stagiaires), cela produit une nouvelle version du logiciel L, disons L'

Questions

- quelle est la licence de L'?
- qui sont les ayants droit de L'?
- qui peut changer la licence de L'?
- quel impact sur les choix de valorisations de L'?

D'où plusieurs approches … cela meriterait un cours entier

- CLA: contributor license agreement
- DCO: developer certificate of origin

Outline

Economie du logiciel (libre)

Breaking news: rapport sur le logiciel libre en Europe (09/2021)!

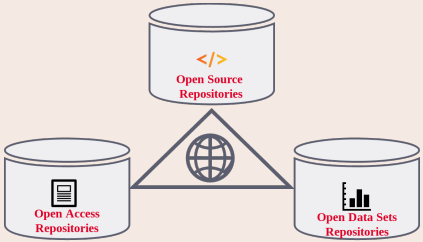
- https://openforumeurope.org/open-source-impact-study/
- "open source software contributes between €65 to €95 billion to the European Union's GDP and promises significant growth opportunities for the region's digital economy"

- cela necessiterait un cours entier à part
- on peut regarder quelques exemples si vous le souhaitez (slides à part)

Outline

Software Source code: pillar of Open Science

Three pillars of Open Science



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

What about FAIR?

FAIR data principles *for data*

in a nutshell: metadata, metadata, metadata all over the place

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 multinationals

...

let's focus on more actionable properties: ARDC is a good starting point

Outline

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

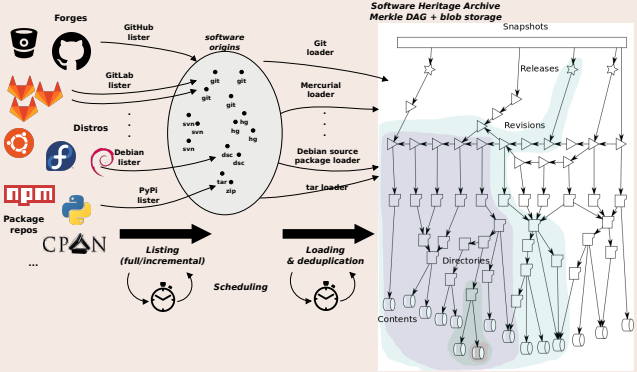
preserve all software source code

Research infrastructure

enable analysis of all software source code

A(rchive)

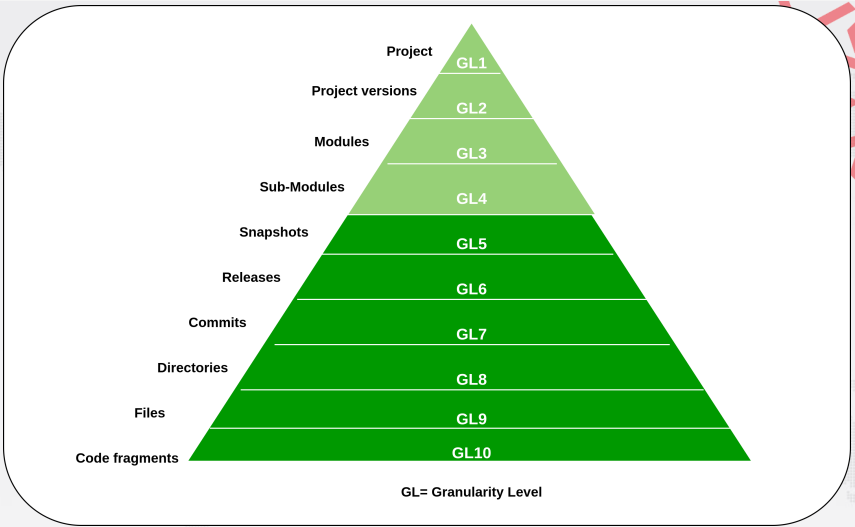
Universal source code archive *not only research* (9B+ files, 150M+ projects)



- 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

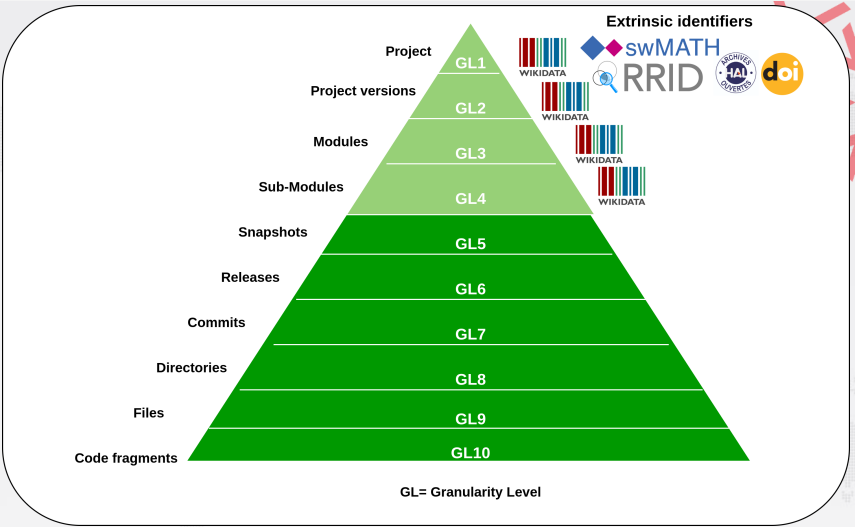
10.15497/RDA00053



Top concept layers vs. bottom artifact layers

R(eference): granularity and identifiers

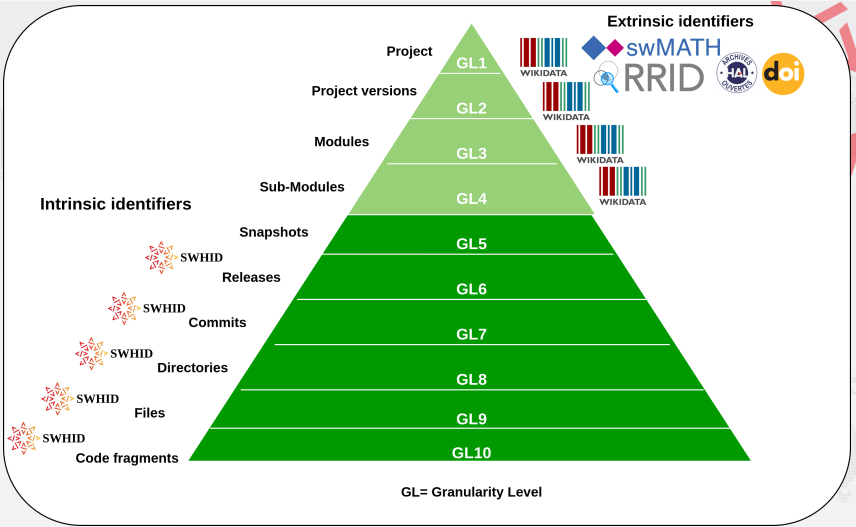
10.15497/RDA00053



Top concept layers vs. bottom artifact layers

R(eference): granularity and identifiers

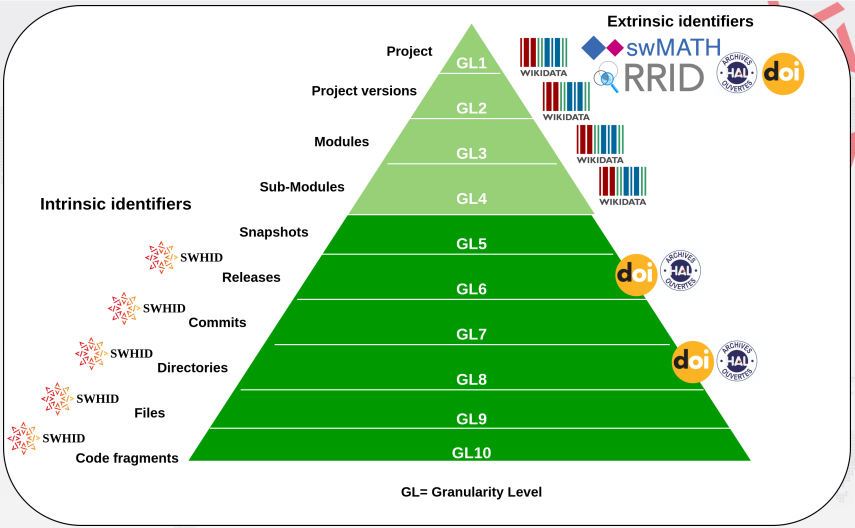
10.15497/RDA00053



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

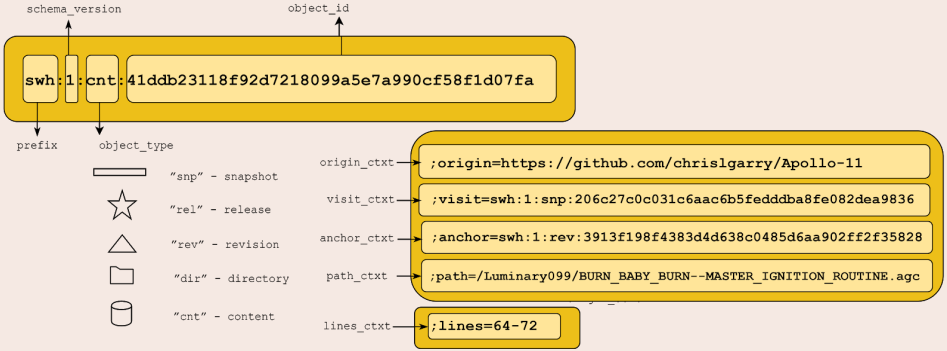
more in this dedicated blog post (with pointers to literature)

Meet the SWHID intrinsic identifiers

Software Heritage Identifiers (SWHID)

link to full docs

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

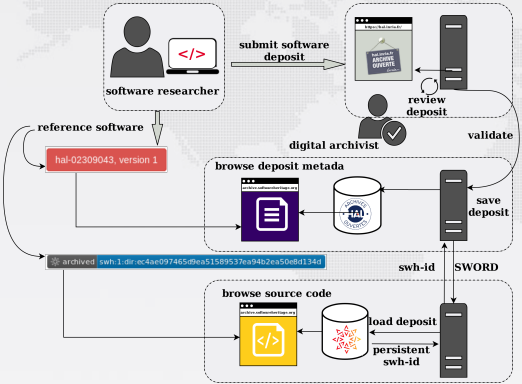


Emerging standard : Linux Foundation SPDX 2.2; IANA registered; WikiData P6138

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

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

National Plan Open Science

Policy: Europe

EOSC SIRS report

- SWHIDs
- archive

Guidelines

- summary
- ICMS 2020

Outline

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)
- 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

Outline

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

Sustainability

- economic model
- community and governance
- license

Evaluation and profit sharing

- contributor roles and relevance

Technical

- infrastructure, tools, processes, quality assurance

> A license is not a business model, a forge is not a community
>
> Cedric Thomas, OW2 CEO

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

Outline

A unique opportunity

Let's work together to build the software pillar of Open Science

Thank you!

Questions?

Appendix

Outline

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources.
The license shall not require a royalty or other fee for such sale.
Rationale: By constraining the license to require free redistribution, we eliminate the temptation to throw away many long-term gains in order to make a few short-term sales dollars. If we didn't do this, there would be lots of pressure for cooperators to defect.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form.
Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge.
The source code must be the preferred form in which a programmer would modify the program.
Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.
Rationale: We require access to un-obfuscated source code because you can't evolve programs without modifying them. Since our purpose is to make evolution easy, we require that modification be made easy.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

Rationale: The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

Rationale: Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations. Accordingly, an open-source license must guarantee that source be readily available, but may require that it be distributed as pristine base sources plus patches. In this way, "unofficial" changes can be made available but readily distinguished from the base source.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

Rationale: In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore we forbid any open-source license from locking anybody out of the process. Some countries, including the United States, have export restrictions for certain types of software. An OSD-conformant license may warn licensees of applicable restrictions and remind them that they are obliged to obey the law; however, it may not incorporate such restrictions itself.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

Rationale: The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. We want commercial users to join our community, not feel excluded from it.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

Rationale: This clause is intended to forbid closing up software by indirect means such as requiring a non-disclosure agreement.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

Rationale: This clause forecloses yet another class of license traps.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

Rationale: Distributors of open-source software have the right to make their own choices about their own software.

Yes, the GPL is conformant with this requirement. Software linked with GPLed libraries only inherits the GPL if it forms a single work, not any software with which they are merely distributed.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

Rationale: This provision is aimed specifically at licenses which require an explicit gesture of assent in order to establish a contract between licensor and licensee. Provisions mandating so-called "click-wrap" may conflict with important methods of software distribution such as FTP download, CD-ROM anthologies, and web mirroring; such provisions may also hinder code re-use. Conformant licenses must allow for the possibility that (a) redistribution of the software will take place over non-Web channels that do not support click-wrapping of the download, and that (b) the covered code (or re-used portions of covered code) may run in a non-GUI environment that cannot support popup dialogues.