



Use Cases and Software Source Code Identification

SCID WG

research data sharing without barriers
rd-alliance.org

27th March 2020 - RDA 15th Plenary Virtual Meeting

Contributors (alphabetical order by name)

- Alice Allen, Astronomy Source Code Library & U. Maryland, USA
- Anita Bandrowski - University of California San Diego, USA
- Roberto Di Cosmo - Software Heritage, Inria and University of Paris, France
- Martin Fenner - DataCite, Germany
- Morane Gruenpeter - Inria, Software Heritage, France
- Daniel S. Katz - University of Illinois at Urbana-Champaign, USA
- John Kunze - California Digital Library, University of California, USA
- Moritz Schubotz - swMATH, FIZ Karlsruhe, Germany

- Introduction
 - Objectives, Outputs, Connections to other WG
- Use cases (Archiving, Referencing, Describing, Citing, etc.)
 - Audience may suggest additional use cases
- Identifiers schemas
 - DOIs, Hashes, SWH-ID, Wikidata entities, ARKS, ASCL-ID, RRID, swMATH-ID
- Small group discussion
 - documenting the use case : challenges, pros, and cons of different identifiers per use case
- Report-back and discussion
- Wrap-up

Introduction

Software Source Code Identification Working Group

Co-chairs

- Roberto Di Cosmo
- Martin Fenner
- Daniel S. Katz

Web page

<https://www.rd-alliance.org/groups/software-source-code-identification-wg>

Repository

<https://github.com/force11/force11-rda-scidwg>

Chronology...

Spawned at RDA **P11** in Berlin from

- the RDA SSC IG &
- the FORCE11 SCIWG

10/2018 - TAB endorsement

4/2019 - RDA **P13**, Philadelphia

- **WG kick-off**
- **ASCL & SWH presentation**

10/2019 - **FORCE2019**, Edinburgh

[Full day hackathon](#) on research software

- Bring together people involved/interested in software identification
 - Talk about why this is an issue
 - Talk about different types of identifiers
 - Talk about use cases
 - Discuss pros and cons of different identifiers for different use cases
 - Document discussions
- Produce concrete recommendations for the academic community

Expected outputs

Medium-term goals (M12)

- An initial collection of software identification **use cases** and software **identifier schemas**.
- An overview of the different contexts in which software artifact identification is relevant, including:
 - Scientific **reproducibility**
 - Fine grained **reference** to specific code fragments from scientific articles or documentation
 - Description of **dependency** information
 - Citation of software projects for proper **credit attribution**

Long-term goals (M18)

- Call out other RDA groups, in particular those working on citation and versioning issues, for consultation on the draft guidelines
- A set of guidelines for persistent software artifact identification, in each of the above contexts

Related Project - FORCE11 Software Citation

FORCE11 Software Citation Implementation Working Group

(co-chairs: N. Chue Hong, M. Fenner, D. S. Katz)

Following-on from FORCE11 Software Citation Working Group and the [Software Citation Principles](#) it developed

Objective: Produce concrete guidelines for software citation, and implement them within the scholarly research community (software developers, repositories and registries, journals and conference and publishers, indexers, institutions)

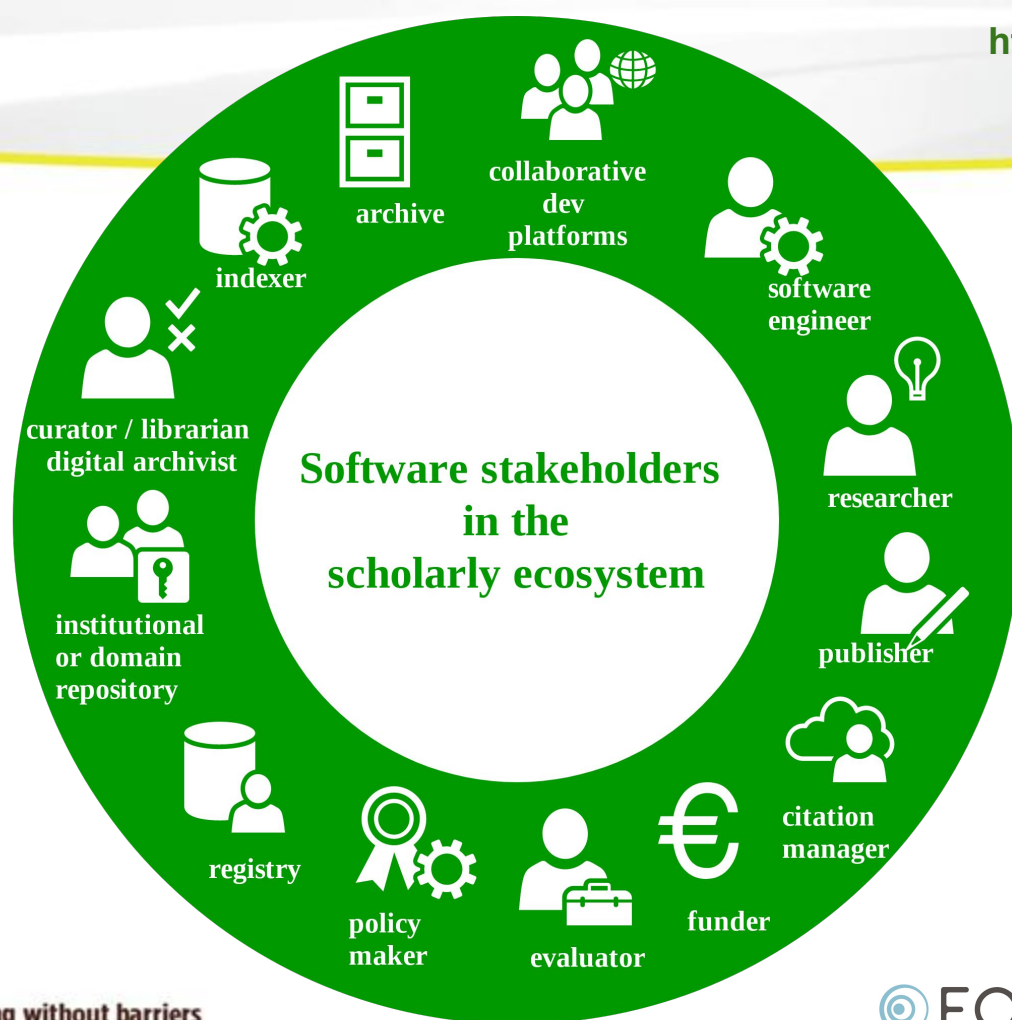
A community with monthly calls to discuss challenges and progress in implementing software citation, with task forces for

- **CodeMeta** - standardizing metadata for software, moving towards merging into schema.org
- **Guidance** - developing documents for developers, authors, and reviewer
- **Journals** - coordinating editors and publishers to simplify and implement guidance
- **Repositories** - developing best practices document for handling software

Use cases

Actors

<https://tinyurl.com/qpg7n7m>



What is at stake

[Archive]

ensure (research) software artifacts *are not lost*

[Reference]

ensure (research) software artifacts *can be precisely identified*

[Describe]

make it easy to *discover / find* (research) software artifacts

[Credit]

ensure *proper credit* is given *to authors*

researcher :

- **access and use** SSC *no longer available* on a collaborative platform [**Archive**]
- **reproduce** an experiment detailed in an article (replication studies) [**Reference**]
- **reference** SSC used in an article (SageMath algorithm example) [**Reference**]
- **search and find** appropriate SSC using rich metadata [**Describe**]
- **give and get credit** for research SSC via correct *citations* to articles and data [**Credit**]

Let's see some concrete examples

Use case: replication studies

A *researcher* wants to reproduce* an experiment from an article [Archive] [Reference]

See the [10 years Reproducibility challenge](#):

1. **find the source code**
2. make small modifications
3. **run and reproduce**
4. **write** (reproducible) report and share

Here is a detailed example:

[Reproducing and replicating the OCamlP3I experiment](#)

TEN YEARS REPRODUCIBILITY CHALLENGE

RESCIENCE SPECIAL ISSUE
FREE TO READ - FREE TO PUBLISH

Workshop
June 22, 2020
BORDEAUX



Would you dare to run the
code from your past self ?

(the one that does not answer mail)

SUBMISSION DEADLINE 01/04/2020

<http://rescience.github.io/ten-years>

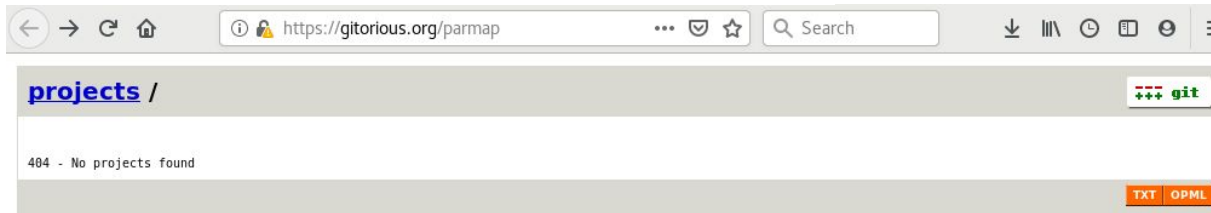
In association with Inria, CNRS, Software Heritage, ReScience, Comité pour la Science Ouverte,
URFIST Bordeaux & Mission de la pédagogie et du numérique pour l'enseignement supérieur.
Contact: nicolas.rougier@inria.fr

Use case: (re)use a software tool

A *researcher* wants to **access and use** SSC presented in an article, that might no longer be available on a collaborative development platform [**Archive**]

6. Conclusions

Parmap is a minimalistic library allowing to exploit multi-core architecture for OCaml programs. It has been designed with the goal of providing parallel map and reduce to OCaml programmers in a fairly natural way, such that the “minimal disruption” principle stated by Cole in his skeleton manifesto paper is enforced. In fact, in order to use Parmap, it is sufficient to substitute the calls to List functions with calls to the equivalent Parmap functions. The clean and efficient implementation of Parmap is such that nearly optimal speedups are achieved on state-of-the-art multi-core architectures when suitable grain computations are parallelized. The full source code of the Parmap library is available under the LGPL licence from <http://gitorious.org/parmap>, and is now also incorporated in the GODI installation system for OCaml librairies.



Use case: link to the *algorithm*

Researcher references in an article the exact relevant code fragment [Reference]

```
1 let simplemapper ncores compute opid al combine =
2   (* init task parameters *)
3   let ln = Array.length al in
4   let chunksize = ln/ncores in
5   (* create descriptors to mmap *)
6   let fdarr=Array.init ncores (fun _ -> tempfd()) in
7   (* spawn children *)
8   for i = 0 to ncores-1 do
9     match Unix.fork() with
10    0 -> (* children code: compute on the chunk *)
11         (let lo=i*chunksize in
12          let hi=if i=ncores-1 then ln-1
13                else (i+1)*chunksize-1 in
14          let v = compute al lo hi opid in
15          marshal fdarr.(i) v;
16          exit 0)
17    | -1 -> failwith "Fork error"
18    | pid -> ()
19  done;
20  (* wait for all children *)
21  for i = 0 to ncores-1 do ignore(Unix.wait()) done;
22  (* read in all data *)
23  let res = ref [] in
24  (* accumulate the results in the right order *)
25  for i = 0 to ncores-1 do
26    res := ((unmarshal fdarr.((ncores-1)-i)):'d)::!res
27  done;
28  (* combine all results *)
29  combine !res;;
```

[Clickable link](#)
(click here!)

Software Heritage

Archive

Archive Access

Browse

Web API

Features

Search

Vault

Save code now

Miscellaneous

Help

```
101 let simplemapper ncores compute opid al collect =
102   (* flush everything *)
103   flush_all();
104   (* init task parameters *)
105   let ln = Array.length al in
106   let chunksize = ln/ncores in
107   (* create descriptors to mmap *)
108   let fdarr=Array.init ncores (fun _ -> tempfd()) in
109   (* call the GC before forking *)
110   Gc.compact ();
111   (* spawn children *)
112   for i = 0 to ncores-1 do
113     match Unix.fork() with
114     0 ->
115         begin
116           let lo=i*chunksize in
117           let hi=if i=ncores-1 then ln-1 else (i+1)*chunksize-1 in
118           let exc_handler e j = (* handle an exception at index j *)
119             info "error at index j=%d in (%d,%d), chunksize=%d of a total of %d got exception"
120               j lo hi chunksize (hi-lo+1) (Printexc.to_string e) i;
121             exit 1
122           in
123           let v = compute al lo hi opid exc_handler in
124           marshal fdarr.(i) v;
125           exit 0
126         end
127     | -1 -> info "fork error: pid %d; i=%d" (Unix.getpid()) i;
128     | pid -> ()
129   done;
130   (* wait for all children *)
131   for i = 0 to ncores-1 do
132     try ignore(Unix.wait())
133     with Unix.Unix_error (Unix.ECHILD, _, _) -> ()
134   done;
135   (* read in all data *)
136   let res = ref [] in
137   (* iterate in reverse order, to accumulate in the right order *)
138   for i = 0 to ncores-1 do
139     res := ((unmarshal fdarr.((ncores-1)-i)):'d)::!res;
140   done;
141   (* collect all results *)
142   collect !res
143 ;;
```

Figure from [a real research article](#)

Corresponding code fragment

Use case: highlight software fragments

A *journalist* links to telltale fragments of the Apollo11 SSC [Reference][Archive]

The code that took America to the moon was just published to GitHub, and it's like a 1960s time capsule July 9, 2016 By Keith Collins

QUARTZ

EMAILS EDITIONS B

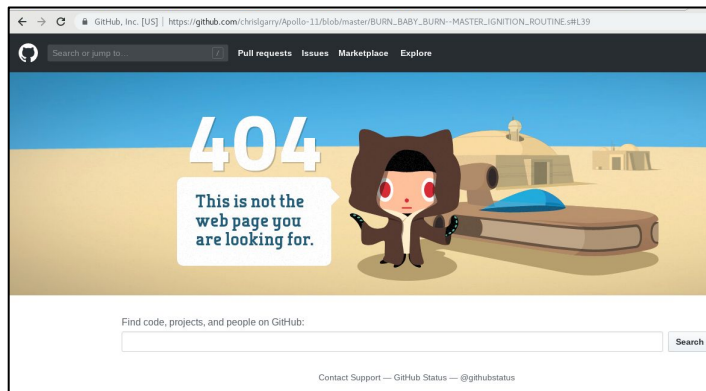
One of the source code files, for example, is called `BURN_BABY_BURN - MASTER_IGNITION_ROUTINE`, and the [opening comments](#) explain why:

About 900 lines into that [subroutine](#), a reader can see the playfulness of the original programming team come through, in the first and last comments in this block of code:

In the file called `LUNAR_LANDING_GUIDANCE_EQUATIONS.s`, it appears that [two lines of code](#) meant to be temporary ended up being permanent, against the hopes of one programmer:

In the same file, there's also [code](#) that appears to instruct an astronaut to “crank the silly thing around.”

“That code is all about positioning the antenna for the LR (landing radar),” Burkey explained. “I presume that it’s displaying a code to warn the astronaut to reposition it.”



[Software Heritage blog post with all archived references](#)

Important remark: *relevant SSC may not be author archived!*

Use case(s): career and activity reports

- **Researcher:** curriculum vitae, promotion, activity report, grant applications
- **Lab/team:** track software production and contributions
- **University/Research institution:** tech transfer, metrics, scientific policy

Granularities can be **coarser** than a release:

“Inria created OCaml and Scikit-learn”

Contributions can be more fine grained than just “author/contributor”:


Architecture, Management, Development, Documentation, Testing, ...

More use cases



- software engineer :
 - **contribute** and improve existing SSC [**Contribute**]
- digital archivist :
 - **browse** the development history of legacy SSC (Apollo11 example) [**Archive**]
- funder :
 - **identify and evaluate** the impact of the funded software projects [**Describe**][**Credit**]
- registry :
 - **identify** and **curate** the software entries I hold [**Archive**] [**Reference**] [**Describe**] [**Cite**]
- Audience may suggest additional use cases [here](#)





Access Parmap example (archived copy)

[Software Heritage guidelines for research software](#)


Software Heritage Archive



Home Development Documentation [Donate](#)



 **Browse archived directory for origin** <https://gitorious.org/parmap/parmap.git> 


 Visits  Snapshot date: 30 March 2016, 07:40 UTC  Branches (29)  Releases (0)

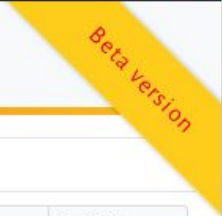
Branch: HEAD ▾ 06c287c /

Go back in history 

File	Mode	Size
 .gitignore	-rw-r--r--	15 bytes
 MOVED-TO-GITHUB	-rw-r--r--	91 bytes




 History  Actions ▾

 Permalinks

Beta version

Access and identify a point in history

Software Heritage Archive



Home Development Documentation

Donate

Beta version

Permalinks

Browse archived revisions history for origin <https://gitorious.org/parmap/parmap.git>

Visits Snapshot date: 30 March 2016, 07:40 UTC Branches (29) Releases (0)

Branch: HEAD Actions

sort by: ☒ revision date ☐ DFS ☐ DFS post-ordering ☐ BFS

Revision	Author	Date	Message	Commit Date
b9boecd	Roberto Di Cosmo	17 July 2012, 12:21 UTC	Moved to github	17 July 2012, 12:21 UTC
ec7a234	Roberto Di Cosmo	17 July 2012, 08:24 UTC	Added code to handle empty input sequences	17 July 2012, 08:24 UTC
0a78aad	Roberto Di Cosmo	14 June 2012, 21:47 UTC	Made stdout/stderr redirection optional (default: false)	14 June 2012, 21:47 UTC
2469b21	Roberto Di Cosmo	13 June 2012, 21:28 UTC	Fixing redirection bug	13 June 2012, 21:28 UTC
103	Roberto Di Cosmo	06 June 2012, 13:03 UTC	Added default ncores parameter	06 June 2012, 13:03 UTC
3838e4b	Roberto Di Cosmo	06 June 2012, 13:03 UTC	Make sure we call List.map if nproc <= 1	06 June 2012, 13:03 UTC
096692	Roberto Di Cosmo	06 June 2012, 11:39 UTC	Removed debugging code from mandels_sdl.ml and added command...	06 June 2012, 11:39 UTC
69b61a2	Roberto Di Cosmo	04 June 2012, 17:33 UTC	Merge branch 'sdl'	04 June 2012, 17:33 UTC
29b2778	Roberto Di Cosmo	20 May 2012, 18:21 UTC	New version of Mandelbrot example using SDL.	04 June 2012, 17:32 UTC
https://archive.softwareheritage.org/browse/revision/ec7a2341ac3d9d8b571bbdfb90a089d4e54dea56/?origin=https://gitorious.org/parmap/parmap.git	levels of the lib...	31 May 2012, 11:41 UTC		

Ice-breaker: propose your use case

<https://tinyurl.com/qpg7n7m>

Identifiers schemas

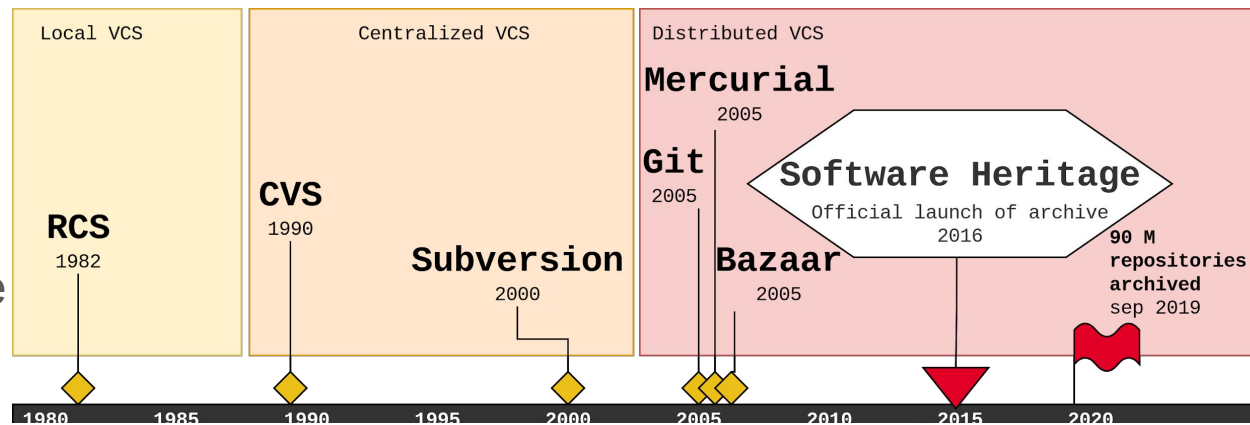
Cryptographic Hashes in VCS

Version control system (VCS)

- records changes made to a (set of) source code file (s)
- allows to operate on versions: diff/merge/fork/recover etc.
- essential tool for software development

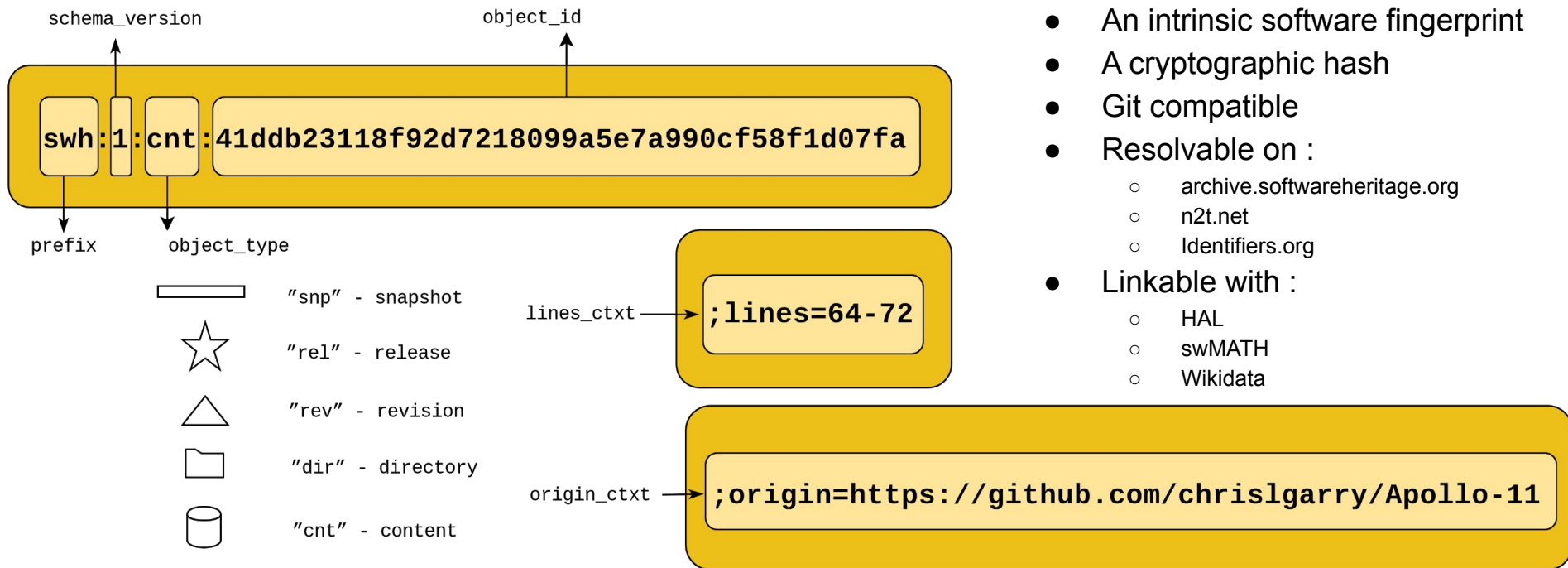
Guarantees:

- Unique identification
- Artifact integrity
- Work for tree structure



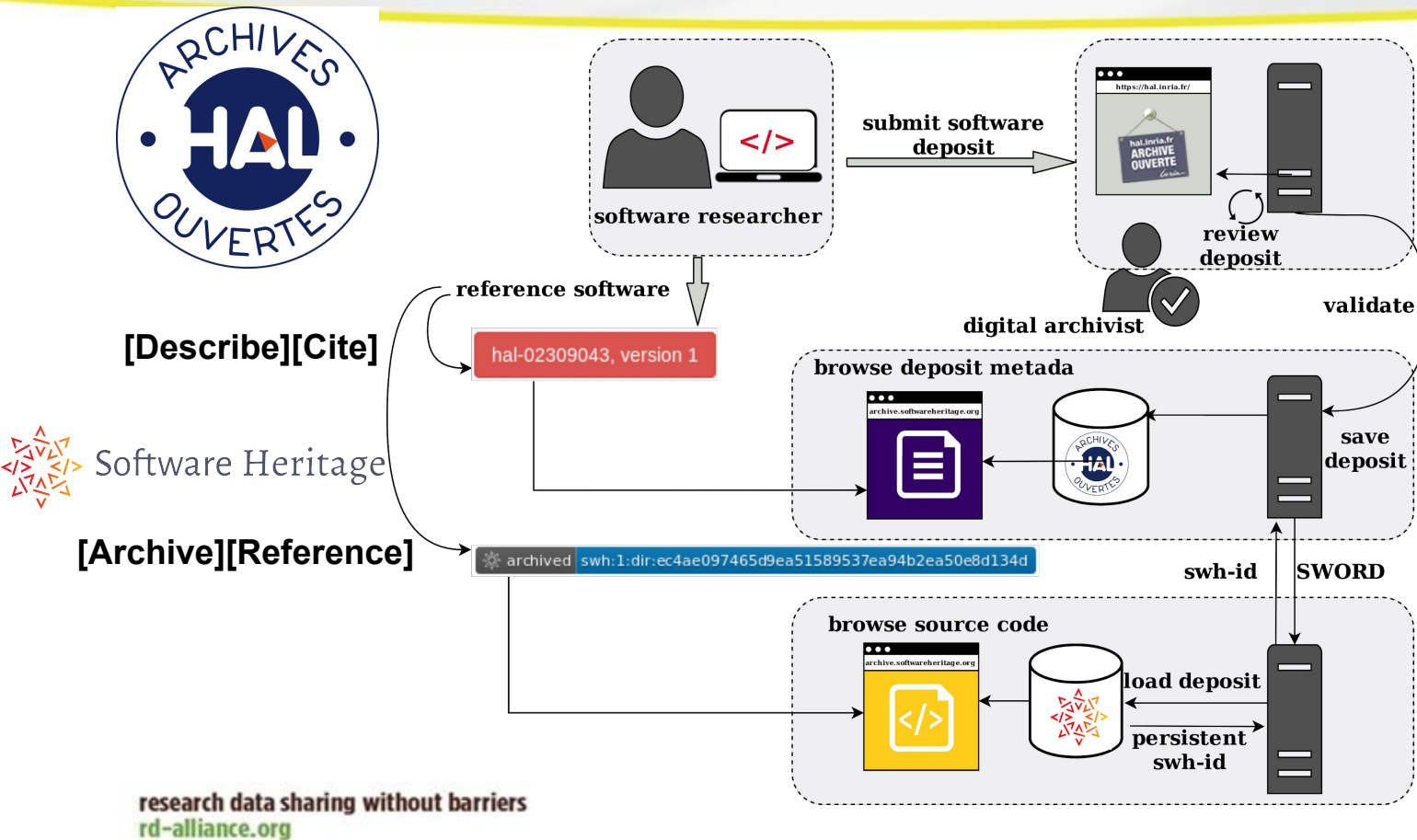
Software Heritage ID ([more info here](#))

[Archive][Reference]



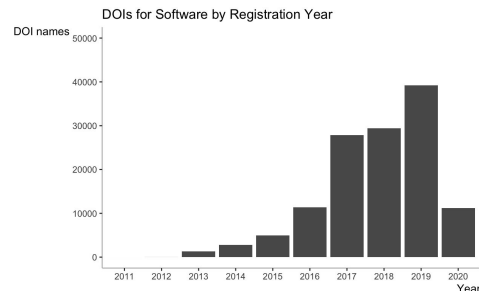
Deposit software on the french national archive HAL

[Deposit guide](#)



DOI (Digital Object Identifier)

Persistent identifier supporting standard citation metadata, and linking to other PIDs. DataCite has registered 128,276 DOIs (84% in Zenodo) for software as of March 26, 2020.



Cited, e.g. in bioRxiv preprint <https://doi.org/10.1101/534834>:

- [14] A. Gitter, “Single-cell RNA-seq pseudotime estimation algorithms,” *doi:10.5281/zenodo.1297422*, Jun 2018.

Included in ORCID record, e.g. <http://orcid.org/0000-0002-9247-0530>:

bcbi/ModelSanitizer.jl: v0.3.0

Zenodo

2019-08-06 | other

DOI: [10.5281/zenodo.3361519](https://doi.org/10.5281/zenodo.3361519)

Source: DataCite

★ Preferred source (of 4)

ARK (Archival Resource Key)



With no fees, 3.2 billion ARKs have been assigned by 580 institutions to things digital, physical, & abstract; resolution is decentralized or, via n2t.net, centralized

Assigners choose (e.g., legacy commit ids) or generate (e.g., opaque ids such as UUID or Noid) name strings, which are registered with redirection target URLs

Example: `ark:/12345/b67c89d/part3.cvs`, where **12345** is the institution, **b67c89d** the thing, `/part3` optional subthing, and `.cvs`, optional variant qualifier

Cite in actionable form, eg, `n2t.net/ark:/12345/f98g76`; ARKs appear in the Data Citation Index, Wikipedia, Wikidata, ORCID profiles, and Internet Archive

Registries identifiers: ASCL-ID

Astrophysics Source Code Library: Registry and repository for source code in astrophysics

Items registered by authors (or sometimes journal editors or users) or added by ASCL editors based on their appearing in the astrophysics literature

Identifiers are `ascl:yymm.xxx`, where *yy* & *mm* are year & month of addition to ASCL, and *xxx* indicates that software was the *xxx*'th ASCL entry in the month

ASCL is indexed by the [SAO/NASA Astrophysics Data System](#) (ADS) and Web of Science; entries can be [cited](#) using their unique ASCL identifier

Registries identifiers: RRID

Research Resource Identifiers are used mostly in biomedicine, registered via [SciCrunch](#): a system that aggregates ~25 RRID registries or repositories, such as the antibody registry, or Addgene repository

The SciCrunch registry is a listing of software projects (e.g., SPSS, ImageJ), services (e.g., core facilities), and data projects (e.g., NeuroMorpho.org) that may need to be cited as **aggregate entities** in the scientific literature, but authors may not wish to cite a specific bit of source code

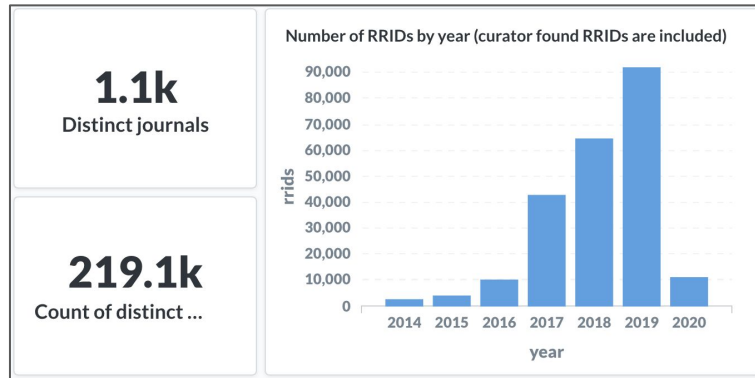
RRID format: **RRID:SCR_12345** (*SCR* = repository code, *12345* = local identifier in that repository)

Why register? Journals ask authors to do so.

Where are they? Methods sections (table of reagents) ->



The screenshot shows a search result for the RRID:SCR_003070. The search bar at the top contains the text "RRID:SCR_003070". Below the search bar, it indicates "Page 2 of about 1,250 results (0.04 sec)". The main content area displays a search result for a paper titled "Comparing the Use of Research Resource Identifiers and Natural Language Processing for Citation of Databases, Software and Other Digital Artifacts" by CN Hsu, A Bandrowski, TH Gillespie, et al. The snippet below the title explains the RRID format: "... that oversees a particular type of resource. For example, 'RRID:SCR_003070' is a syntax RRID for the software tool ImageJ, and 'SCR' is the prefix of the SciCrunch Registry. Supplemental Table S1 provides a list ...".



Registries identifiers: swMATH-ID

- swMATH provides information on software referenced in mathematical publications
- Software is identified with a numeric Identifier, e.g., 825=SageMath
- In addition it provides informations on
 - Authors
 - Classification of the software
 - Information on citations in mathematical Publications
- The dataset is manually curated
- Back and forth linking efforts with
 - Wikidata
 - Software Heritage

The screenshot displays the swMATH website. At the top, there is a search bar with the swMATH logo and navigation links for 'Search', 'Advanced search', and 'Browse'. Below the search bar, the entry for 'SageMath' is shown. It includes a description of SageMath as free, open-source math software, a list of authors (The Sage Developers, William Stein, David Joyner, David Kohel, John Cremona, Éric Cal, Burçin), and a link to the manual. A section titled 'Keywords for this software' contains a word cloud with terms like 'Galois representations', 'modular forms', 'rational points', 'Gröbner basis', 'Young tableaux', 'cryptanalysis', 'elliptic curve', 'matroid', 'superficial value', and 'minimum rank'. Below this, there is a list of references in zbMATH, sorted by year (citations), showing results 1 to 20 of 96. The references list works by Adamar, Alhajjar, Garcia, Ghosh, Grace, Haraway, Bahsoun, Bettin, Evoniuk, Brandfonbrener, Dahlberg, Evoniuk, Fredi, Grace, Gyárfás, and others. On the right side, there are links to 'URL: www.sagemath.org', 'Code', 'InternetArchive', 'Manual', and 'Add information on this software'. At the bottom right, there is a section for 'Article statistics & filter:' with a search bar, a 'Clear' button, and a table showing the MSC classification of the articles. The table has columns for 'MSC classification / top' and 'Top MSC classes', with rows for '05 Combinatorics', '11 Number theory', '14 Algebraic geometry', '20 Group theory and...', and '68 Computer science'. Below the table, there is a 'Publication year' filter with checkboxes for '2010 - today', '2005 - 2009', '2000 - 2004', and 'before 2000'. At the very bottom right, there is a 'Chart: cumulative / absolute' showing a bar chart of publications over time.

Wikidata entities

- Numeric identifiers prefixed with Q, e.g, [Q1165184](#)=SageMath
- Version information is maintained with the property software version identifier [P348](#)
- Identifier can be merged to [remove duplicates](#)
- Open editing
- Multilingual
- All kinds of “external” Identifiers, e.g,
zbMATH Work ID, [P6830](#),
Twitter username, [P2002](#)...



Main page
Community portal
Project chat
Create a new item
Create a new Lexeme
Recent changes
Random item
Query Service
Nearby
Help
Donate

Tools

What links here
Related changes
Special pages
Permanent link
Page information
Concept URI
Cite this page

Item Discussion Read

Sage (Q1165184)

mathematical software application
System for Algebra and Geometry Experimentation | SageMath

[edit](#)

▼ In more languages
Configure

Language	Label	Description	Also known as
English	Sage	mathematical software application	System for Algebra and Geome... SageMath
German	Sage	Computeralgebrasystem	
French	Sage	logiciel mathématique	sagemath
Bavarian	No label defined	No description defined	

All entered languages

Statements

Instance of	<div><div><div><div></div><div>free and open-source software</div></div><div><div></div><div>▼ 0 references</div></div></div><div>edit</div></div>
	<div><div><div><div></div><div>computer algebra system</div></div><div><div></div><div>▼ 0 references</div></div></div><div>edit</div></div>
	<div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>add reference</div></div>
	<div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>add value</div></div>

logo image

sage

[edit](#)

Wikipedia (21 entries) [edit](#)

ca	SAGE (programari matemàtic)
cs	Sage (software)
da	SageMath
de	Sage (Software)
el	Sagemath
en	SageMath
es	SageMath
fi	Sage (ohjelmisto)
fr	SageMath
hi	सेजमैथ
id	Sage
it	Sage (software)
ja	SageMath
ko	SageMath
nl	Sage (software)
pl	Sage (system algebry komputerowej)
pt	SageMath
ro	Sage
ru	Sage
sr	SageMath
zh	Sage

Wikibooks (0 entries) [edit](#)

Wikinews (0 entries) [edit](#)

Identification target - what do we want to identify?

Taken from: [Identifiers and targets crosswalk](#)

Software artifact

- Executable (download link)
- Software source code
 - Dynamic artifact - current development code (on collaborative development platform)
 - Archived copy
 - Release / Package
 - Commit / a specific point in development history
 - Directory
 - File / algorithm

Software concept / project / collection

- Description in registry
- Homepage

Software context

- Complementary artifacts
- Data
- Articles
- Documentation

Ice-breaker v2: propose the identification target for a use case (as a comment)

<https://tinyurl.com/qpg7n7m>

Discussion in small groups

1. Introduce yourself to your neighbours
(name, affiliation, a software identification use case that interest you)
2. Choose group use case
 - You can choose a use case from the list in the use cases [directory](#) or propose a new use case
3. Document use case
 - Make a copy of use case template in use cases [directory](#)
4. Analyse use case
 - Pros for each identifier schema
 - Cons for each identifier schema
5. Discuss which identifiers are most relevant for the particular use-case

Use case: Title

Use case summary									
Actor/s				Step by step scenario • •					
Goal									
Use case extensions				Examples • •					
Target for identifier Metadata record / software source code artifact / software executable (with/without container) / other: _____				Granularity level (bold selection) project / collection / repository / branch / release / commit / directory / file / lines of code					
Identifiers schemas and examples									
ASCL	ARK	DOI	HAL	Hash	RRID	SWH	SwMath	Wikidata	Other:

Thanks