Use Cases and Software Source Code Identification

SCID WG

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
Agenda

- 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

https://tinyurl.com/qpg7n7m
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
Goals

● 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

Output due June 2020

https://tinyurl.com/qpg7n7m
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
Software stakeholders in the scholarly ecosystem

- Archive
- Collaborative development platforms
- Software engineer
- Researcher
- Publisher
- Citation manager
- Funder
- Evaluator
- Policy maker
- Registry
- Institutional or domain repository
- Curator/librarian/digital archivist
- Indexer

Partly taken from the Software Citation Principles
https://peerj.com/articles/cs-86/
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*
Use cases

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
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 libraries.
Use case: link to the algorithm

**Researcher** references in an article the exact relevant code fragment [Reference](#)

```
let simplemapper ncores compute opid al combine =
  (* init task parameters *)
let ln = Array.length al in
let chunksize = ln/ncores in
(* create descriptors to mmap *)
let fdarr=Array.init ncores (fun _ -> tempfd()) in
(* spawn children *)
for i = 0 to ncores-1 do
  match Unix.fork () with
  0 -> (* children code: compute on the chunk *)
    (let loc=i+chunksize in
     let hi=(i+1)+chunksize-1 then ln-1
     else (i+1)+chunksize-1 in
     let v = compute al to hi opid in
     marshal fdarr.(i) v; exit 0)
  | -1 -> failwith "fork error"
  | pid -> ()
  done;
  (* wait for all children *)
for i = 0 to ncores-1 do ignore(Unix.wait()) done;
(* read in all data *)
let res = ref [] in
(* accumulate the results in the right order *)
for i = 0 to ncores-1 do
  res := ((unmarshal fdarr.(i)(ncores-1-i))):!res
  done;
(* combine all results *)
combine !res;
```

Figure from a real research article

Corresponding code fragment
Use case: highlight software fragments

A journalist links to telltale fragments of the Apollo11 SSC

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

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.”

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](https://tinyurl.com/qpg7n7m)
## Access Parmap example (archived copy)

### Software Heritage guidelines for research software

- **Browse archived directory for origin** [https://gitorious.org/parmap/parmap.git](https://gitorious.org/parmap/parmap.git)

<table>
<thead>
<tr>
<th>File</th>
<th>Mode</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>.gitignore</td>
<td>-rw-r--r--</td>
<td>15 bytes</td>
</tr>
<tr>
<td>MOVED-TO-GITHUB</td>
<td>-rw-r--r--</td>
<td>91 bytes</td>
</tr>
</tbody>
</table>
Access and identify a point in history

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Author</th>
<th>Date</th>
<th>Message</th>
<th>Commit Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec7a234</td>
<td>Roberto Di Cosmo</td>
<td>17 July 2012, 08:24 UTC</td>
<td>Added code to handle empty input sequences</td>
<td>17 July 2012, 08:24 UTC</td>
</tr>
<tr>
<td>b9b0e6d</td>
<td>Roberto Di Cosmo</td>
<td>17 July 2012, 12:21 UTC</td>
<td>Moved to github</td>
<td>17 July 2012, 12:21 UTC</td>
</tr>
<tr>
<td>e74213e</td>
<td>Roberto Di Cosmo</td>
<td>14 June 2012, 21:47 UTC</td>
<td>Made stdout/stderr redirection optional (default: false)</td>
<td>14 June 2012, 21:47 UTC</td>
</tr>
<tr>
<td>8726b7d</td>
<td>Roberto Di Cosmo</td>
<td>13 June 2012, 21:28 UTC</td>
<td>Fixing redirection bug</td>
<td>13 June 2012, 21:28 UTC</td>
</tr>
<tr>
<td>e5b97b6</td>
<td>Roberto Di Cosmo</td>
<td>13 June 2012, 21:28 UTC</td>
<td>Added default ncores parameter</td>
<td>13 June 2012, 21:28 UTC</td>
</tr>
<tr>
<td>69b61a7</td>
<td>Roberto Di Cosmo</td>
<td>04 June 2012, 17:33 UTC</td>
<td>Make sure we call List.map if nproc &lt;= 1</td>
<td>04 June 2012, 17:33 UTC</td>
</tr>
<tr>
<td>29b778</td>
<td>Roberto Di Cosmo</td>
<td>04 June 2012, 17:33 UTC</td>
<td>New version of Mandelbrot example using SDL.</td>
<td>04 June 2012, 17:33 UTC</td>
</tr>
</tbody>
</table>
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)

- An intrinsic software fingerprint
- A cryptographic hash
- Git compatible
- Resolvable on:
  - archive.softwareheritage.org
  - n2t.net
  - Identifiers.org
- Linkable with:
  - HAL
  - swMATH
  - Wikidata

swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa

prefix object_type

object_id

"snp" - snapshot
"rel" - release
"rev" - revision
"dir" - directory
"cnt" - content

lines_ctx

;lines=64-72

origin_ctx

;origin=https://github.com/chrislgarry/Apollo-11
Deposit software on the French national archive HAL

[Describe][Cite]
hal-02309043, version 1

[Archive][Reference]
archived swh:1:dir:ec4ae097465d9ea51589537ea94b2ea50e8d134d

software researcher
submit software deposit
review deposit
validate
digital archivist
save deposit
swh-id
SWORD
browse deposit metadata
load deposit
persistent swh-id
browse source code

Software Heritage
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:


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

Slide credit: Moritz Schubotz, SwMath
**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`, ...

Slide credit: Moritz Schubotz, SwMath
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
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

<table>
<thead>
<tr>
<th>Use case: Title</th>
</tr>
</thead>
</table>

### Use case summary

<table>
<thead>
<tr>
<th>Actors</th>
<th>Step by step scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Goal</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Use case extensions

<table>
<thead>
<tr>
<th>Target for identifier</th>
<th>Granularity level (bold selection)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata record / software source code artifact / software executable (with/without containers) / other: _______</td>
<td>project / collection / repository / branch / release / commit / directory / file / lines of code</td>
</tr>
</tbody>
</table>

### Identifiers schemas and examples

<table>
<thead>
<tr>
<th>ASCL</th>
<th>ARK</th>
<th>DOI</th>
<th>HAL</th>
<th>Hash</th>
<th>RRID</th>
<th>SWE</th>
<th>DataNest</th>
<th>Wikidata</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Discussion in small groups

https://tinyurl.com/qpg7n7m
Thanks