2020

jupytercon

Daniel S. Katz
Stephan Druskat

Software Publication & Citation

1 - Introduction - Why publish your software (part 1)

Introduction

Welcome to this segment of the Open Source Fundamental Tutorial, on Software Publishing and Citation

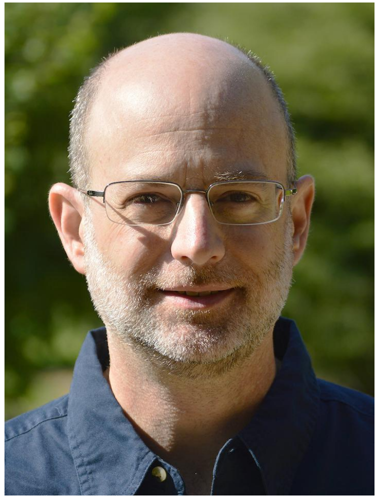
We're:

Daniel S. Katz

University of Illinois at Urbana-Champaign

Assistant Director for Scientific Software & Applications, NCSA

Associate Research Professor, CS & ECE & iSchool



Stephan Druskat

German Aerospace Center (DLR) & Humboldt-Univ. zu Berlin & Friedrich Schiller Univ. Jena

Research Software Engineer, PhD candidate Computer Science

Why publish your software

In an academic context, the most important reasons are:

1. It helps make the research results that have been produced using your software **more reproducible**
   - The published software can be more easily referenced from original research results report
   - People who want to reproduce the results will be able to identify, find, obtain, and run the same software more reliably
   - Published software becomes part of the provenance record for the research
2. Publishing your software makes it possible for you to **receive credit** for your work on the software, especially if you publish it in a way that allows researchers to easily find metadata that they can use to **cite your software**

2 - Why publish your software (part 2)

More reasons to publish your software

Trust

- Publication process is an additional step after software development
- Indicates that the developers are serious about sharing their software
- And likely serious about other aspects of the software too
- Trust especially increased if publication process includes peer review
  - Via documented code reviews
  - Or through more formal peer review
    - Software is published in a software journal that reviews both software and paper

More reasons to publish your software

Quality

- Publication process itself makes developers more conscious about the quality of their software
    - Whole world will be able to see it, use it, and critique it
- Review process (if any) provides means by which the developers get feedback about their practices
    - And can improve them

More reasons to publish your software

Publisher requirements

- Publisher (of research results) may require software ("artifact") to also be published
- Not all academic publishers require publication of software (and data) at this time
- Some journals are trying to increase reproducibility
  - Have started asking for code submissions, for example using tools such as CodeOcean

More reasons to publish your software

Visible impact

- Publishing your software helps to make the impact of software on research more visible
- Clear today that software is used in almost all research disciplines
- When researchers cite published software - and are required to do so - more and more software contributions to research become visible
- This helps people who create or maintain this software receive credit for their software work

3 - How to publish your software (part 1)

Two main routes to publish software

1. Treat software like a paper or preprint (author-like)

2. Treat software like software (developer-like)

Software publication: author-like (1)

- We have a long history of publishing papers (lots of publishers)
  1. Author submits paper and metadata to publisher (e.g. Wiley, AAS)
  2. Publisher reviews paper
  3. Publisher archives accepted paper, provides identifier (typically Digital Object Identifier (DOI))
- We have also mostly accepted publishing preprints (fewer but still many publishers)
  1. Author submits preprint and metadata to publisher (e.g., arXiv, BioRxiv)
  2. Publisher archives preprint, provides identifier (e.g., arXiv ID, DOI)

Software publication: author-like (2)

- Can treat software like preprints
  1. Author submits software and metadata to platform such as Zenodo
     - Metadata includes author information, software name, version identifier, publication date, and the location where the software can be found
  2. Platform archives software, provides identifier (e.g., DOI)
- "Software Citation Principles" paper encourages this to treat software as a first class citizen in research, like papers and preprints

But software is not a paper

Open source software is not really developed like papers

- Model for papers
  - Authors work on paper
  - Decide it is done and they want to share it
  - They publish it
  - Others can see it, and cite it
- Model for software
  - Developers work on software in the open
  - Others can see it, use it, contribute to it at any time
  - No publication step
  - Ideally any version can be cited at any time

Software publication: developer-like (1)

Software Heritage

- UNESCO project aiming to build the universal software archive
- Collects source code from open repositories such as GitHub, GitLab, Bitbucket and others, and saves it in a distributed long-term archive

Similar to how the Internet Archive, archive.org, saves web content

- Lets you identify and view versions of web content that no longer exist

Software publication: developer-like (2)

Software Heritage automatically archives software code

- You can also request that a specific repository be archived
- Archived software remains citable, even if original repository (or even platform) vanishes

Software Heritage does not provide DOIs for archived software

- Instead, it provides its own persistent identifiers (SWHIDs), at multiple levels of resolution
  - For a whole repository
  - For a release or a single commit
  - For specific directories and file contents
  - For specific lines of code in a source code file

Software Heritage is great for archiving and identifying software source code

- But for academic citation, also need citation-relevant metadata
- Software Heritage doesn't help with this
- We'll show ways to do this later

Software publishing - pros and cons

Author-like software publishing

- Publish your software and correct and complete metadata on a platform that gives you a DOI for it, such as Zenodo
- A good choice now to make your software part of your scholarly record

Developer-like software publishing

- To give people a way to identify and reference highly specific versions or parts of your software, depositing it in the Software Heritage archive
- Focuses on the software, but may not be captured in your scholarly record today

4 - How to publish your software (part 2)

Software publishing - other options (1)

Version control

If: You are not interested in academic credit, or you cannot publish your software with a DOI
Then: put your software in public version control - the most basic step to make your software available
Example: Push your code to an open repository on GitHub, GitLab, Bitbucket, or another public coding platform

Comments:

- This makes it hard to cite your software in a paper
- It doesn't qualify as publication in the traditional sense - there's no metadata available to cite your software for example, unless you provide it
- Users will need to figure out the exact version they want to use by themselves
- The software is not really preserved (archived) - you can remove it later, or the platform you are using may not survive, e.g. as happened with Google Code
- Automatic archiving (e.g., by Software Heritage) should eventually preserve (archive) your software
- But this is a valuable first step - better than doing nothing

Software publishing - other options (2)

Software paper

If: You want academic credit, and you like traditional papers, which are easy to cite
Then: Write a paper about the software concept, algorithm, or functionality; submit it to a traditional journal
Example: List of journals that may accept your paper:
https://www.software.ac.uk/which-journals-should-i-publish-my-software

Comments:

- This is not really software publication; software itself is not published, may not be peer-reviewed
- When you finish the next version of the software, may not be feasible or possible to write new paper for it, especially if you follow "release early and often" good software engineering practice, or if users want to cite a version between software papers
- A reason to choose this: You can't share the code, maybe your institution does not allow it, or the code is protected for legal reasons, or is system-critical or safety-critical

Software publishing - other options (3)

Software journal paper

If: You want to publish the software, have it reviewed alongside a paper, and want something easy to cite
Then: Write a paper and submit it and the software to a software paper journal
Example: The Journal of Open Source Software (JOSS)

Comments:

- A compromise between publishing just the software and more traditional paper publication
- Software journals have arisen recently, and focus on the publication of research software.
- Includes a paper about the software (possibly short) and software metadata, references, etc.
- Review, preservation, and identifier is for both the paper and code
- Review of software may include: quality, implementation of best practices, documentation and tests, reproducible environments, etc., and also on the domain aspects of the software
- Only a specific version of the software is published and reviewed
- This is still not fully software publication

Software publishing - other options (4)

Software indices

If: You want to cite someone else's software, or want an identifier for all versions of your software
Then: Submit it to a software index
Example: There are indices for institutions, groups of institutions, and discipline-specific software, such as swMath, for mathematical software, bio.tools for bioinformatics software, SciCrunch for software from the life sciences, and the Astrophysics Source Code Library (ASCL) focuses on software developed for astrophysics research

Comments:

- These indices also provide persistent identifiers for their entries
- Different journals may or may not accept these identifiers/citations
- Most indices allow adding software that was not developed by the people who add it software to the index
- You can also use Software Heritage to achieve something similar: you can request the archiving of a source code repository, and once it's archived use the provided Software Heritage identifier to reference the software

5 - How to publish your software (part 3)

Author-like software publishing - details (1)

"Software Citation Principles" recommends traditional DOI-based publication route for actual software:

1. You provide the source code of your software as well as its metadata to a long-term archival repository
2. The source code and metadata are archived in the repository, which returns a DOI that points to the software and includes the metadata
3. Usually, both are provided together on a landing page

Important aspects that make this suitable for software publication and citation:

1. Software metadata includes all relevant information needed to cite software, e.g., its name, complete list of authors, version information, and publication date
2. Software source code can include documentation and other artifacts related to the software
3. Can publish all release versions of the software in this way

Author-like software publishing - details (2)

- To link between different published versions of software, you could use a separate DOI, with metadata pointing to the DOIs of the different versions
  - In theory, this could be done manually, for example to link between versions published with different publishers, but there doesn't exist any tooling for this yet, other than via citation
  - Thankfully, some repositories such as Zenodo provide this "parent" DOI automatically for all versions that are published there
- This is currently the preferred/recommended way to publish software:
  - It publishes the actual software version, not a text proxy
- Issues:
  - No peer-review - this must be done outside of the publication process
    - Via code reviews of new source code added to the source code repository, or as post-publication audits
  - Only archived releases can be cited

6 - Why make your software citable

Software publications and citation metadata

- Some publication forms ensure the existence of metadata
  - author-like publication
  - Publication in software journals
- What about the completeness and correctness of metadata?
  - When is this asserted?
    - Before, during, or after publication? At the time of citation?
  - How is this asserted?
    - Reviews? Technical editing? Research? Asking?
    - Correct author metadata is hard to assert unless you’re the author!
  - Who asserts this?
    - Software authors? Reviewers and editors? Authors citing software?

Define and provide correct and complete metadata for your software!

Reasons to make your software citable

1. You and others can track usage and impact
2. Quantifiable impact may further your career
3. Promote the status of software in academia
4. Lead by example

7 - How to make your software citable (metadata about software) (part 1)

What do people need from you to be able to cite your software?

They need **metadata!**

What metadata do people need?

1. They want to cite the software concept
   - To compare it to other software
   - To discuss problems it solves, algorithms it uses

2. They want to cite a version of the software
   - To cite the version they've used

	Concept	Version
Name	✓	✓
Authors	✓	✓
Location	✓	✓
Version info		✓
Publication date	?	✓

Author metadata

- There may be more authors than code committers
- Not everyone who has committed code may be an author
- Only authors can determine and record who is an author
- Author information may change between versions
    - Authors join the project
    - Previous authors' code may be deleted/replaced

Where to store metadata?

- A lot of metadata is already stored with the source code
- Author metadata should also be stored there
- Metadata should be updated when code is updated
- Metadata should be committed to version control

8 - How to make your software citable (metadata about software) (part 2)

Metadata files and formats

- Store software metadata in source code repository
  - Make it readable for humans and machines
  - Enables citation of all versions
- Two best options:
  - File in the Citation File Format (CFF) - `CITATION.cff`
    - Only citation metadata (for software and dependencies = references)
    - Human- and machine-readable (YAML)
    - Can contain guidance for users ("If you want to cite this software, use these metadata")
    - Compatible with BibTeX, RIS, EndNote, CodeMeta, schema.org, .zenodo.json
    - Tools available for generation, conversion
  - CodeMeta file - `codemeta.json`
    - Generic format for software metadata, more than just citation metadata
    - Machine-readable (JSON-LD), arguably human-readable

Plain-text metadata files

Another option: plain-text `CITATION` (or `CITATION.md`) file

- Only human-readable
- May include wrong citation style
- May include wrong citation format (what if you don't use BibTeX?)

Better to include machine-readable file and tooling to convert to formatted citation

- Possible today for `CITATION.cff` via BibTeX for *TeX documents

Citation File Format

https://citation-file-format.github.io

```
cff-version: 1.1.0
message: "Please cite this software using these metadata."
authors:
  - family-names: Druskat
    given-names: Stephan
    orcid: https://orcid.org/0000-0003-4925-7248
  - name: "The MRS contributors"
title: "My Research Software"
version: 2.0.4
doi: 10.5281/zenodo.1234
date-released: 2017-12-18
```

Other fields:

- Repository URLs
- Version control information
- Other identifiers
- Keywords
- Description
- License information
- References (incl. dependencies)

Generation:

- Via web form, manually

Conversion:

- Via web service, tooling

CodeMeta

https://codemeta.github.io

```
{
  "@context": "https://doi.org/10.5063/schema/codemeta-2.0",
  "@type": "SoftwareSourceCode",
  "identifier": "https://doi.org/10.5281/zenodo.1234",
  "name": "My Research Software",
  "version": "2.0.4",
  "author": [
    {
      "@type": "Person",
      "givenName": "Stephan",
      "familyName": "Druskat",
      "@id": "https://orcid.org/0000-0003-4925-7248"
    },
    {
      "@type": "Project",
      "name": "The MRS contributors"
    }
  ],
  "datePublished":"2017-12-18"
}
```

Other fields:

- Repository URLs
- Version control information
- Other identifiers
- Keywords
- Description
- License information
- References (incl. dependencies)
- ... and many more

Generation:

- Via web form, tooling

Conversion:

- Via tooling

9 - When and how to cite software?

When to cite software?

You should cite software whenever
good scientific practice says you should!

- When you have used it for research you publish, and your results depend on the software
- When it is part of the state-of-the-art / related work (or a similar text section)
- When you use it in your own software

And perhaps in your CV!

How to cite software?

It depends on where you want to cite:
text document, software, dataset?

- Text: Create a formatted citation from the metadata

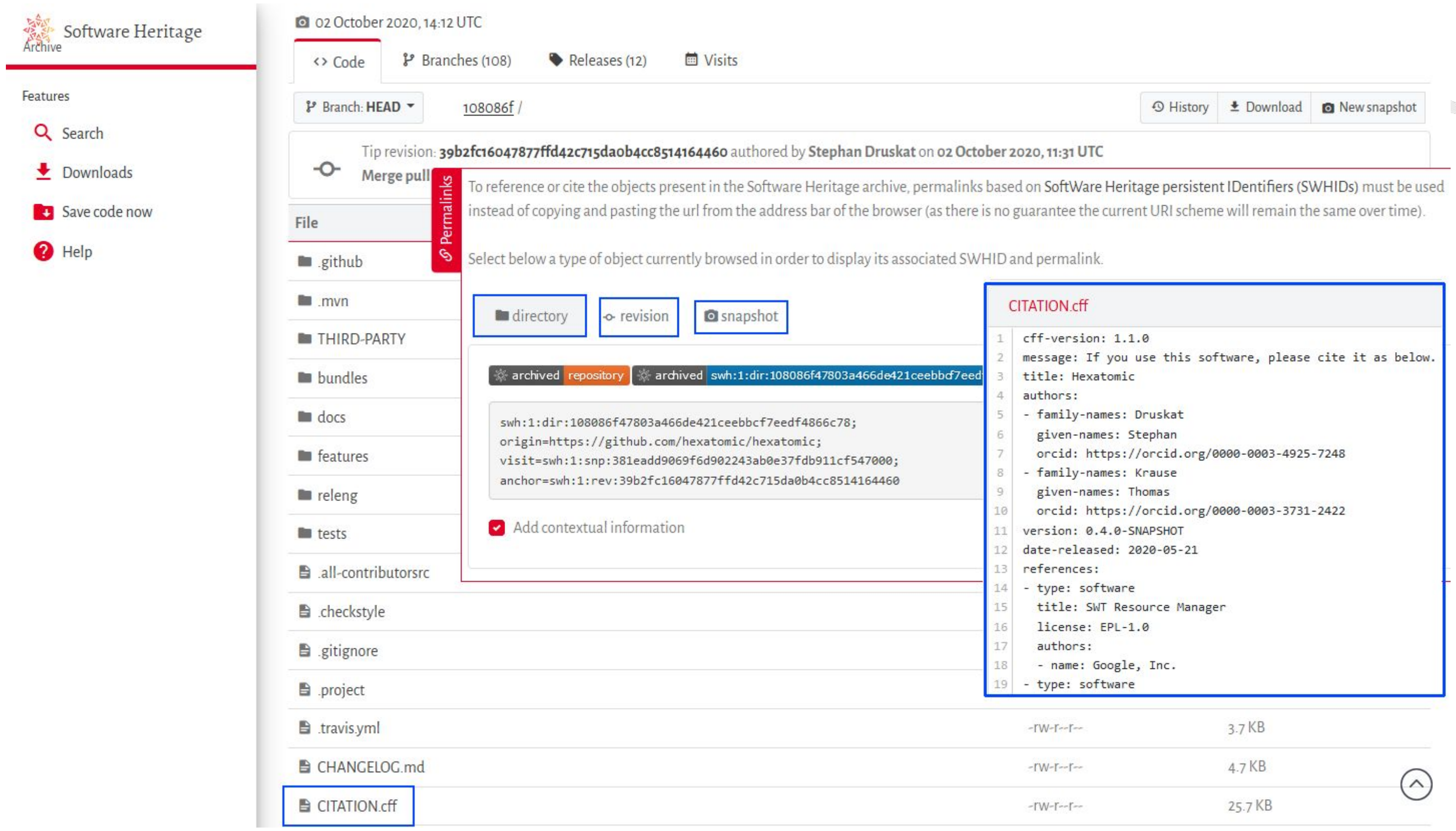
- Software/dataset: add cited software metadata to product metadata

    - If you cite software A in software B, add software A metadata to metadata file for software B, e.g.,
        - in the `references` section in `CITATION.cff`, or
        - the `citation` section in `codemeta.json`

Where to find citation metadata?

- Publication repository (Zenodo, etc.): use the DOI/PID and associated metadata
- Software paper: Cite the software paper
  - If the version in the paper is another than the one you used: also cite the software version itself!
- Index: The index identifier and provided metadata
- Source code repository: metadata file
- Software Heritage:
  - The Software Heritage identifier for the whole repository
    + an archived metadata file, or
    find out name and use “<Name> project” as author
  - The Software Heritage identifier for a specific version
    + an archived metadata file, or
    find out name and use “<Name> project” as author

Software Heritage identifiers



Tip revision: **39b2fc16047877ffd42c715da0b4cc8514164460** authored by **Stephan Druskat** on **02 October 2020, 11:31 UTC**

Merge pull

To reference or cite the objects present in the Software Heritage archive, permalinks based on SoftWare Heritage persistent IDentifiers (SWHIDs) must be used instead of copying and pasting the url from the address bar of the browser (as there is no guarantee the current URI scheme will remain the same over time).

Select below a type of object currently browsed in order to display its associated SWHID and permalink.

```
swh:1:dir:108086f47803a466de421ceebbcf7eedf4866c78;
origin=https://github.com/hexatomic/hexatomic;
visit=swh:1:snp:381eadd9069f6d902243ab0e37fdb911cf547000;
anchor=swh:1:rev:39b2fc16047877ffd42c715da0b4cc8514164460
```

✓ Add contextual information

File		
.github		
.mvn		
THIRD-PARTY		
bundles		
docs		
features		
releng		
tests		
.all-contributorsrc		
.checkstyle		
.gitignore		
.project		
.travis.yml	-rw-r--r--	3.7 KB
CHANGELOG.md	-rw-r--r--	4.7 KB
CITATION.cff	-rw-r--r--	25.7 KB

CITATION.cff

```
cff-version: 1.1.0
message: If you use this software, please cite it as below.
title: Hexatomic
authors:
- family-names: Druskat
  given-names: Stephan
  orcid: https://orcid.org/0000-0003-4925-7248
- family-names: Krause
  given-names: Thomas
  orcid: https://orcid.org/0000-0003-3731-2422
version: 0.4.0-SNAPSHOT
date-released: 2020-05-21
references:
- type: software
  title: SWT Resource Manager
  license: EPL-1.0
  authors:
  - name: Google, Inc.
- type: software
```

Citing unpublished software

- Archive it on Software Heritage,
  cite using an SWH identifier
- Add it to an index,
  cite using the identifier

- Both: metadata will be incomplete or imprecise, find name and use “<Name> project” as author

Further guidance

Further guidance on software citation

- Software Citation Principles paper (doi:10.7717/peerj-cs.86): Outlining the principles of software citation (author-like publication)

- Software citation checklists for
    - software developers (doi:10.5281/zenodo.3482769)
    - paper authors (doi:10.5281/zenodo.3479199)

- cite.research-software.org - High-level overview of software citation, practical suggestions on how to
    - cite software
    - make software citable

thanks