Practical Software Citation for Research Software Developers, Maintainers and Users
Date: 4 June 2025
Presented by: Stephan Druskat (German Aerospace Center (DLR), Institute of Software Technology)

(The slides are available via the link in the page's sidebar.)

Q: CFF question: the date-released is for the current version or for the initial release, or the initial commit ??

A: `date-released` is for the current version (i.e., the latest release). Whenever you are planning to make a release, the value should be updated to reflect the actual release date.

Q: There are websites for publishing data, e.g. https://radar.kit.edu/radar/en/dataset/zYYmSEIWMwoELRCK#. Is it a good idea to use these resources? Of course we already know about github for publishing codes.

Audience notes: Google Code: Why GitHub is not a publication platform

A: Generally, there is nothing wrong with publishing software artifacts to repositories that have been originally set up for data. But there are some prerequisites:

- The repository should agree that it's okay to publish software there as well (there may be documentation/guidelines around this)
- The repository should provide a metadata schema that supports software publications. This means that:
    1. It should provide at least one specific `type` for *software* (where type is the type of publication, e.g., article, thesis, data, …)
    2. It should provide the fields that you'd want to see for the publication of a software version (importantly, a `version` field)
    3. It should provide *software* licenses (in addition to text/data licenses, such as Creative Commons licenses) to choose from. CC licenses aren't suitable for software, and hence you want to be able to pick the software license for your software for the publication. Examples include Open Source Initiative-approved licenses.

4. It should ideally provide some way of linking different versions of the same software to reflect that these are versions of the same “thing”, not different things. This could be the possibility to record DataCite relations between versions in the metadata, or the creation of one identified set of metadata for the project (e.g., the *concept DOI in Zenodo*, whose metadata contains a list of all versions (using the DataCite relation `hasVersion` (example)).

To Michael’s comment on Google Code: Yes, the disappearance of Google Code is one very good example why making your source code publicly available is NOT the same as *software publication*. In the case of Google Code, the Software Heritage Archive thankfully stepped in

To Alfred’s mention of a specific (data) repository: It does already contain software (see, e.g., this version of openCARP), and the openCARP team have also built a tool to automate publication on RADAR (somewhat similar to HERMES). However, you can see that this repository doesn’t seem to fully support software publications. If you compare the metadata with the Zenodo metadata for a version of another software:

Metadata	Zenodo	RADAR
Resource type (`resourceTypeGeneral`)	Software	Software
Resource type declared on website	Software	Dataset
License (see `rightsList`)	OSI-approved, SPDX-identified software license (Apache-2.0)	1. “Open Access” (not applicable to software) 2. “Other” (lists Academic Public License on website, not in SPDX license list)
Version identifier (e.g., 1.4.2, 16.0)	Recorded in `version` field	Not recorded (inserted into title)
Version linking	Yes (via metadata field and DataCite relation `isVersionOf`)	No, each version is treated as separate publication

I think you can see why not having the correct metadata schema can be problematic for human users, and machines alike. If I wanted to identify the software publications in a repository, and how they are connected, I’d have a better time with one repo than with the other.

Q: What are the other good software journals (besides JOSS), especially for OSS?

Audience notes: SoftwareX (Elsevier's public software journal), JOSS is great, for imaging - , pyOpenSci also peer reviews software (and partners with JOSS) to provide a paper: https://www.pyopensci.org/, domain-specific journals such as ACM Transactions on Mathematical Software

A: The Software Sustainability Institute maintains a list of journals where you can publish software (and also where you can "publish software"). Some of them do actual reviews of the software itself, some accept papers about software. I'll highlight

- For open source software: Journal of Open Research Software (specifically their Software Metapapers track)
- For image processing software: IPOL Journal - Image Processing Online

I still stick to my argument that "software journals" - and particularly the outstanding JOSS, the Journal of Open Source Software - do great work in solving some problems in software publication (organizing peer review, supporting the archival and identification process, providing a "semantic bridge" in calling themselves journals (which is what management understands)), but they are unfit (by design) to solve others (regularly publishing versions). Some aspects they help with, but are fundamentally not ideal, e.g., highlighting the importance of software by writing a descriptive paper (however short) that has to go with the source code. As such, I think software journals are a great "bridge technology", but the optimum lies somewhere between software journals and repository-based publishing. The one big issue we need to solve is peer review, i.e., quality assurance for each version.

Q: What is your experience and advice for citing contributors whose contributions are often not captured by git commits, such as those who contribute to requirements, analysis, and design?

Audience notes: Is https://credit.niso.org/ (CRedIT Taxonomy) relevant?

A: There are different options for this. The AllContributors specification and tooling focuses on the issue of non-code-pushing contributors specifically. They cover many roles for community work, ideation, etc. IMHO, you can't go wrong with using these specs to document contributions in the source code repo.

On another level, it should be possible for contributors of "non-git-committers" to qualify for authorship. This is why it is important for research software projects to think about authorship, who qualifies for it, the criteria, etc. As mentioned, we're collaborating in the ReSA Task Force Software Authorship & Contribution to develop guidance around software authorship, and will invite the community to review it and provide input.

For more “formally” recognizing and acknowledging contributors in addition to authors, e.g., in publication metadata, it should be possible to record them in a specified field (the next version of the Citation File Format will include a `contributors` field for this, CodeMeta also includes such a field via Schema.org), and these metadata should also be recorded in the publication repos. E.g., Zenodo does have a field “contributors” backed by a controlled vocabulary, but (just like the CRedIT taxonomy mentioned above) is high-level and does not allow you to specify software-specific roles.

Q: How does this apply to dynamic software? My package works on a plugin system. Citing the framework doesn’t give the plugin creator proper credit. Citing all plugins that are used could conceivably lead to hundreds of citations.

Audience note: Some systems based on plugins provide the means for the plugins to report their own preferred citations, for example as a command-line option.

A: Good question. There is an underlying question here about how we define and understand software as an object, and where we identify its boundaries. Is something “in” software A or not “in” software A? Are dependencies part of “the software”? Do we talk about source code, binaries, or runtime? Etc.

I think that this is ultimately not a question that “software citation” needs to solve, it’s an issue of the academic system as such. Why is hundreds of citations a problem? Technically it’s not (we don’t count them by hand, PDFs can hold many citations and aren’t expensive, etc.). It may become a problem (for the system of citation, credit, metrics, etc.) where we don’t trust the citation, or rather, the citation intent and intention by the citer.

Beyond this philosophical discussion, I don’t see plugins as much of a problem. Either they’re developed “apart” from the framework, then they should have their own citation metadata, and be cited when used (unless you discuss the whole ecosystem around the framework, in which case you may want to cite all hundreds of plugins). “Apart” meaning: in another source code repository, and/or by another set of authors or a subset of the framework authors, and/or with another roadmap, and/or with separate documentation, etc. When they’re only technically plugins (i.e., implemented to interface with the framework via an extension mechanism), but otherwise developed alongside the framework, they may be a part of the framework (and share the same citation metadata). I’ve seen and done both.

For the concrete case of the question: If a researcher uses, e.g., your and another plugin within the framework, they *should* cite your plugin and the other plugin, they *may* cite the framework if it is relevant (e.g., for their usage of the plugins, and fo reproducibility, etc.), but they *shouldn't* cite all plugins available for the framework.

Q: Good places to publish codes and data have to persist–meaning that the platforms should not disappear sometime down the road. If the data and code are included in citations, you want them to persist. Have any data and code publishing platforms ever been taken down?

A: Taken down: Not that I know of. Source code repo platforms have disappeared, see Google Code above. Of course, there are no *real* guarantees, and diversifying archival is probably a good idea in these times (and anyway).

For software citation, it is important that the metadata persist. For reproducibility you also need the source code (and other things). Generally I think it's a good idea to

- archive your software in the Software Heritage Archive, together with a citation metadata file (and perhaps other metadata files). This way you can get better citation and better reproducibility, AND
- publish your software releases in a publication repository, AND
- (slightly unrelated) consider mirroring your source code repos on another platform (GitHub/GitLab/Forgejo/gitea, etc.)

Q: I do observe that in R package development, the package starts with a blank template to be filled with with info like Author name and email, Maintainer name and email, etc! Does that make life easier than earlier?

A: Yes, because it makes you think early on in the process about a few aspects of authorship and difference between authors and other roles (here: maintainer). It also gets you started with recording these metadata.

Also no, because you still have to maintain the citation metadata along with your software, and you also have to decide how to provide the metadata for the citation and publication cases, and perhaps syncing metadata.