Open Science, Publications and Code

Melissa Harrison

DOI: 10.4000/books.oep.16187
Publisher: OpenEdition Press, Académie des sciences, EDP Sciences
Place of publication: Paris ; Marseille
Year of publication: 2022
Published on OpenEdition Books: October 13, 2022
Series: Laboratoire d’idées
Electronic EAN: 979-10-365-4562-7

http://books.openedition.org

Electronic reference

The text only may be used under licence CC BY 4.0. All other elements (illustrations, imported files) are "All rights reserved", unless otherwise stated.
Open Science, Publications and Code

Melissa Harrison
EMBL-EBI

I am a publisher and have been a publisher for most of my career, including working at eLife. I now work at EMBL-EBI managing an open access repository. We have over 7.5 million full-text articles and preprints, the COVID-19 preprint subset and some others. The perspectives I have gained from those two different roles are interesting. From the perspective of an open access publisher, one of the problems that we encountered at eLife when trying to encourage responsible behaviours such as the use of software and its citing, is that when you are trying to do all these things, you are asking your authors to do yet another thing. There was the response from authors: ‘How do I cite code’? Quite often we found that one of the issues is that when we were trying to dig out what the code is that people were talking about, they would just give us the citation to an original research article publication instead, and that is not really the point. We need to get access to and promote the actual source code underneath and it is not just about research publications. Software does not fit the usual citation styles and publishers do not know how to tag them or how to deal with them. It is a problem that I have been looking at.

Also, as mentioned, software is not static, so it is not like a research publication where you get your DOI, you publish it and then it is done. It is also not the same as data. These are all problematic issues for us as publishers trying to deal with citing code. One solution that has been very popular within the publishing community is the link between GitHub and Zenodo. Authors of code can quite easily archive their code on to Zenodo and they get a DOI. DOI is very publisher-centric and they understand what this is. We have been using DOIs via Crossref for journal articles for a long, long time and it is easy thing for publishers to recognise, and the code is archived somewhere, so it feels like a winner to everyone in that situation. Then Zenodo can also link back to the GitHub site, so if there are changes people can see what is there.

Using some work from a data scientist at Europe PMC, we mined 7 000 random XML documents we had, and we found 300 000 citations. Of that large number there were 10 citations to Zenodo that we found, using a Zenodo DOI. As Roberto was talking about earlier, having a dedicated archive for software, Zenodo is not that. Of those 10 citations, three were datasets, one was a journal article, and the rest were software.
However, looking at what the publishers called that content when they tagged the citations within their XML format, there was only one that was identified as software and the others were all a mishmash of different things, which is problematic if we are looking for proper credit, attribution and reuse of software.

One of the issues that I have often thought about is that “Zenodo” does not equal “code”, so it is not possible for publishers to say, ‘Okay, fine, there is a Zenodo DOI there, tag it as code’, because it is not that simple. Zenodo holds lots of different things.

Publishers are very familiar with Crossref. It is a publisher organisation. They have DOIs and they craft and use the tools there to validate reference, check them and pull in extra metadata if it is missing, or use it to add DOIs to their publications if authors have not provided them. DataCite is the DOI provider for Zenodo and a lot of publishers are not using the tools of DataCite to crosscheck their references, so if there are Zenodo-based DOIs they are quite often not getting the extra validation steps that would allow the publisher to add in the fact that it is code, data or something else.

Another solution, other than the Zenodo solution, is the Software Heritage solution, as you have heard Roberto talk about. This was the way that eLife went. There was a strong feeling within the team that DOIs were publisher constructs, and they were not necessarily the way forward to reference and archive all the different varieties of software. Software Heritage is archiving all this content from a vast number of different sources. Software Heritage links to specific parts of the code and it is reingesting, creating updates, and it indexes more sources. The other thing key thing for eLife was that we did not have to ask the authors to do the work to create the Zenodo link and archive their code themselves. We could do it for them, and that was a key thing for us because, as you know, my first point was, ‘Oh, yet another thing they are asking me to do’. We felt that we did not want to insist that they did the extra work. We were doing it for them.
In terms of the eLife experience, I am really proud of how many changes we made. Regarding the VoR (version of record) publications that we published over time, eLife is a modest-sized journal compared to all of the content now that I now deal with at Europe PMC, but by the end of 2021 the number of software citations matched the number of VoR publications. That is not to say that every article we published had a software citation – some might have had two or three and some might have had none – but it is a far cry from where we were at the beginning, so this is really great to see. However, this takes time and effort and eLife, because of our mission-driven status, had the resources and time to look into this. We introduced quality control, instructions, and documentation. There was training for our staff and vendors. We introduced automated checks. We wrote a lot of code to make sure all of these things were being discovered, from finding a GitHub link within an article and identifying that, asking the author for the reference to also validating and checking all software references as well. The editorial team, as well as the production team, were putting a lot of effort in at the beginning as well, which was great.

Just going back out to the wider concept again, using the skills at Europe PMC and the archive that we have there, using that same 7 000 XML documents, the random documents we found at the beginning, 15 of them had a software publication type, and these only came from eLife and PeerJ, representing a tiny fraction of the citations we are finding. Having found that, my colleague then just looked at eLife and PeerJ.
publications, and we found 6,205 software references that were identified. Out of the top sources identified, 46% were not available and then it was GitHub and then a variation below that. There was then a long, long tail of unique sources.

One of the things I am very involved with as a publisher is JATS for Reuse, which I still chair. Most publishers that are creating full-text context use XML to produce that and JATS, the Journal Article Tagging Suite, is a standard that most people use. Because it is a standard for so many publishers there is a lot of variety, and it has to be effectively loose so that anyone can use it, so JATS4R is a group that has been developed for reuse. The original group that developed this was comprised of open access publishers, thinking about the interoperability and the mining of all this information from that content.

A Software Citation Working Group worked from 2017 to 2020, and we published our recommendation in November 2021. This recommendation is still relatively new, but I remember that when we first set up the working group, a lot of publishers were saying they did not know how to cite software, that there was not a code or term for them to use. eLife was using one, because we invented it and we just went with it, but we realised that that was a blocker for a lot of publishers. One of the key things about the recommendation therefore is giving the publishers the ability to recognise a reference as software and indicate that in the XML so that that is then clear to anyone who is reusing that or who needs to mine for software references.

The other thing about this recommendation was that it had to be as minimal as possible, because the more complexity you add to these things the harder it is for people to take them on. eLife put a lot of effort into very full-structured references, but it is hard. The next thing is that we wanted a pub-id. That is a well-known ID and an identifier for something that is properly archived. GitHub unfortunately does not fit into that remit, but Zenodo and Software Heritage do. In that subset of articles that we looked at from eLife and PeerJ we found that 202 had a pub-id and they were all to DOIs. Failing at that, the next best thing is to find an external link. We all know that external links are not great. It is much better to have a system identifier within the reference, but failing that, an external link works, and so we found in our subset 2,524 external links. However, the sad thing is that 3,834 had no pub-id or external link, which makes them so much harder for people to find, to reuse and just to investigate on the references.

This leads to my conclusion that there is a long road ahead of us and we recognise that, but a lot of progress has been made. Software citation is gaining traction. The fact that it is one of the pillars here as we speak today shows how important it is and
how important people think it should be. Within the publishing world, the fact that I had such a large group within the JATS4R working group who were interested in creating this recommendation and driving it forward was notable. One of the issues I see for people is which path to take. For instance, at eLife we took the Software Heritage route. There is also Zenodo. I am sure there will be more coming up and different ways of dealing with things, and that does cause problems in itself sometimes. You have to reach that tipping point where people all go down one way, but at the moment we are not quite sure yet.

There are many contributors to all of this. It is not just publishers, it is not just authors, it is not just the researchers. There are so many moving parts here, and a big issue is educating people, from publishers to researchers, and everybody needs the education. We also found that when we were talking to our vendors, we had to explain the production and editorial issues, and the feedback loops are extensive, as well as updating on tooling. So, from a publisher perspective, using DataCite and automating checks between, would help with this cause a lot.

To go a step further

“The hardest thing to overcome to improve software citation probably was the authors’ resistance, when trying to explain to them how important it is to get to the source code and cite that rather than a paper associated with it.”

“Software is very different to data, there are different infrastructures involved and it is a different environment, but I definitely would say that I did learn a lot from the implementation of data citations and applying them to software, even though it is a different type of content.”

References