Software Citations and the ACAT Community

Daniel S. Katz

Assistant Director for Scientific Software and Applications, National Center for Supercomputing Applications (NCSA); Research Associate Professor, Computer Science (CS); Research Associate Professor, Electrical and Computer Engineering (ECE); Research Associate Professor, School of Information Sciences (iSchool); University of Illinois Urbana-Champaign, Urbana, Illinois, 61801, USA

E-mail: d.katz@ieee.org

Abstract. Software is essential for the bulk of research today. It appears in the research cycle as infrastructure (both inputs and outputs, software obtained from others before the research is performed and software provided to others after the research is complete), as well as being part of the research itself (e.g., new software development). To measure and give credit for software contributions, the simplest path appears to be to overload the current paper citation system so that it also can support citations of software. A multidisciplinary working group built a set of principles for software citation in late 2016. Now, in ACAT 2017 and its proceedings, we want to experimentally encourage those principles to be followed, both to provide credit to the software developers and maintainers in the ACAT community and to try out the process, potentially finding flaws and places where it needs to be improved.

1. Introduction

The purpose of this paper is to explain an experiment being performed by the 18th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2017). We believe that software is essential in most research, and in particular, ACAT is focused on work that is commonly implemented in software. In late 2016, a set of software citation principles were published that are intended to promote and guide the use of citations of software, similar to citations of papers. In this paper, we bring together software citations and ACAT, explain how software can be cited in ACAT-related work, and urge that it is cited in the ACAT proceedings and in the future ACATs and related meetings, so that the contributions of software developers will be recognized and rewarded, and the amount and quality of software that is shared will increase.

2. The role of software in research

Two types of evidence demonstrate the increased role and importance of software in today's research. While neither of these is specific to the ACAT community, there is no reason to think that these general trends are not also true in physics research in general, and there is reason to believe that they are even more true for the ACAT community, which includes a strong computational component as shown in its name, "Advanced Computing and Analysis Techniques in Physics Research."

The first type is evidence from surveys, where researchers are asked how important software is to them. Two recent surveys, one of UK academics at Russell Group Universities [1, 2], and

one of members of (US) National Postdoctoral Research Association [3, 4] found that 67% / 63% of respondents said, "my research would not be possible without software," where the two results correspond to the two surveys. 21% / 31% said, "my research would be possible but harder," while just 10% / 6% said, "it would make no difference."

The second type is evidence from papers, where the discussions in any form of software in papers are counted, either manually or via natural language processing and machine learning. An informal scan of 6 months of *Science* in mid-2013 found that about half the papers were software-intensive projects, and most of the other papers also relied on some software. A formal study of 90 randomly selected papers in the biology literature in 2015 found that 80% mentioned software, and that those articles mentioned an average of 4.85 software packages [5]. A more recent study of *Nature* in Jan–Mar 2017 found software mentioned in 32 of 40 research articles, with an average of 6.5 software packages mentioned per article [6].

The software that is used in research can be thought of as part of the research process. Some software can be considered inputs to the process, where it is known in advance of the research, and the research plan relies on it. Some software is developed or improved as part of the research itself. And some software that is produced during a research project is intended to be shared with others who can then use it for their own research. Even if the software developed during research is not produced with the intent of sharing it, some limited sharing might be required by community and publisher and funder efforts toward reproducibility and open science. In this paper, where we discuss software citation, we focus on the software that is shared between researchers, either as inputs or outputs. We call this software-as-infrastructure.

3. Software citation principles

In 2015 and 2016, a FORCE11[1] Software Citation working group developed a set of software citation principles [7]. The group started July 2015, co-led by Arfon Smith and Daniel S. Katz. In September 2015, a working group in the WSSSPE3 workshop that focused on credit and citation, co-led by Kyle E. Niemeyer and Daniel S. Katz decided to join forces with the FORCE11 group, with Niemeyer joining as a co-lead of that group. The group grew to about 60 members, including researchers, developers, publishers, repository developer and maintainers, and librarians.

This group started by recognizing that current citation system was created for papers and books, and to develop a system for software citation, we either need to overload the current system to add software or to completely rework the citation system from scratch. The group decided to focus on the overloading path, as the reworking path seems too difficult. The challenge the group tried to address was not just how to identify software in a paper, but how to identify software used within the research process.

Work was done on GitHub (`https://github.com/force11/force11-scwg`) and on the FORCE11 web site (`https://www.force11.org/group/software-citation-working-group`). The group reviewed existing community practices and developed a set of use cases for software citation. It then drafted a software citation principles document, which started with previously published data citation principles [8], updated them based on software use cases and related work, and further updated them based on working group discussions. This led to a draft that was subject to community feedback and review, as well as feedback at a workshop at FORCE2016 in April 2016. Discussion was generally via GitHub issues, and changes in the document were tracked. In late 2016, the paper and its reviews were published [7]. The paper includes a set of six principles (general statements), use cases (where the principles should apply), and discussion (suggestions on how to apply the principles).

[1] FORCE11 (`https://www.force11.org`) is a community of scholars, librarians, archivists, publishers and research funders that has arisen organically to help facilitate the change toward improved knowledge creation and sharing.

The software citation principles are:

(i) Importance. Software should be considered a legitimate and citable product of research. Software citations should be accorded the same importance in the scholarly record as citations of other research products, such as publications and data; they should be included in the metadata of the citing work, for example in the reference list of a journal article, and should not be omitted or separated. Software should be cited on the same basis as any other research product such as a paper or a book, that is, authors should cite the appropriate set of software products just as they cite the appropriate set of papers.

(ii) Credit and Attribution. Software citations should facilitate giving scholarly credit and normative, legal attribution to all contributors to the software, recognizing that a single style or mechanism of attribution may not be applicable to all software.

(iii) Unique Identification. A software citation should include a method for identification that is machine actionable, globally unique, interoperable, and recognized by at least a community of the corresponding domain experts, and preferably by general public researchers.

(iv) Persistence. Unique identifiers and metadata describing the software and its disposition should persist – even beyond the lifespan of the software they describe.

(v) Accessibility. Software citations should facilitate access to the software itself and to its associated metadata, documentation, data, and other materials necessary for both humans and machines to make informed use of the referenced software.

(vi) Specificity. Software citations should facilitate identification of, and access to, the specific version of software that was used. Software identification should be as specific as necessary, such as using version numbers, revision numbers, or variants such as platforms.

In May 2017, the FORCE11 Software Citation Working Group ended, and a new Software Citation Implementation Working Group started, co-chaired by Neil Chue Hong, Martin Fenner, and Daniel S. Katz. This group has the goal of moving the software citation principles to implementation, with this conference experiment as an example of progress. Those interested in following the new group can join it at `https://www.force11.org/group/software-citation-implementation-working-group`.

4. Applying the principles

In general, the principles work to add a step into the software workflow: publishing a version of software to make it citable. As such, implementing the principles in practice for ACAT can be seen as having two parts:

4.1. Making your software citable

If a developer has software they want people to cite, the first step is to publish that software. If the software is on GitHub, which is common today, the developer can follow the steps in GitHub's guide (`https://guides.github.com/activities/citable-code/`). Otherwise, the developer can submit the software to an archive such as Zenodo or figshare. In either case, the submitter needs to supply the metadata for the publication, including the authors, the title, the version, and possibly any citations that software itself needs to make, such as to software that it uses. Once this is done, the service that accepts the software will provide a DOI. The developer can then create a CITATION file with a suggested citation that includes this DOI, update the README file similarly, or otherwise tell people how to cite the software. The developer can also write a software paper and ask people to cite that, but this is secondary, just since our current system doesn't work well for software.

4.2. Citing someone else's software
The author who wants to cite someone else's software should first check for a CITATION file or a README file. If either file says how to cite the software itself, the author should do that, and if not, the author should do their best to follow the principles. Specifically, they should try to include all contributors to the software, and if this is not clear, they can just name the project. They should try to include a method for identification that is machine actionable, globally unique, interoperable, perhaps as a URL to a release or a company product number, if no DOI is available. If there is a landing page that includes metadata, they should point to that, not directly to the software (e.g., the GitHub repo URL.) They should include specific version/release information. And, if there's a software paper, they can cite this too, but not in place of citing the software itself.

5. Examples

This proceedings uses a citation style that is somewhat unhelpful for software, and that seems focus on compactness at the cost of clarity, which may not be fully appropriate in a time where most papers are distributed electronically, and having DOIs or URLs where a work can be found are probably more important to the reader than saving a few photons. Thus, work is needed to fit software into this style, and the examples that follow should be considered a starting point that others can adapt to better meet the needs of the community. In addition, the `iopart-num` BibTeX package produced by Mark A. Caprio and provided to the proceedings' authors ideally should be updated to include a software type.

Some examples of citations for unpublished software are:

- Geant4 Project 2017 Geant [software] version 10.3.2 Available from `https://github.com/Geant4/geant4/releases/tag/v10.3.2` [accessed 2017-08-17]
- Eigen Project 2017 Eigen [software] version 3.3.4 Available from `https://bitbucket.org/eigen/eigen/` [accessed 2017-08-17]
- Python Project 2017 Python [software] version 3.6.2 Available from `https://www.python.org/downloads/release/python-362/` [accessed 2017-08-17]
- LLVM Project 2017 LLVM Core [software] version 4.0.1 Available from `http://releases.llvm.org/download.html#4.0.1` [accessed 2017-08-17]
- R Project 2017 R [software] version 3.4.1 Available from `https://cran.r-project.org/src/base/R-3/` [accessed 2017-08-17]
- TensorFlow 2017 Project TensorFlow [software] version 1.3.0 Available from `https://github.com/tensorflow/tensorflow/releases/tag/v1.3.0` [accessed 2017-08-17]
- Collobert R, Farabet C, Kavukcuoglu K, Chintala S, Leonard N, Tompson J, Zagoruyko J, Massa F, Dundar A, Jin J, et al. 2017 Torch [software] commit a0bf77ff070ca27eb2de31c6465f8ffa4e399be2 available from `https://github.com/torch/torch7` [accessed 2017-08-17]

And here are some examples of citations of published software:

- Pfenninger S and Pickering B 2017 calliope-project/calliope [software] Release v0.5.2 *Zenodo* `https://doi.org/10.5281/zenodo.810012`
- Heinrich L and Cranmer K 2017 diana-hep/packtivity [software] Initial Zenodo Release *Zenodo* `https://doi.org/10.5281/zenodo.309302`
- Stasto A, Xiao B and Zaslavsky D 2014 SOLO [software] version 1 *figshare* `https://doi.org/10.6084/m9.figshare.1033996.v1`

- Dawe EN, Ongmongkolkul P and Stark G 2017 root_numpy: The interface between ROOT and NumPy *Journal of Open Source Software* **2.16** https://doi.org/10.21105/joss.00307
- Rademakers F, Canal P, Naumann A, Couet O, Moneta L, GANIS G, Vassilev V, Piparo D, Bellenot B, wverkerke, et al. 2017 root-project/root: Release v6-11/02 *Zenodo* https://doi.org/10.5281/zenodo.1003159

6. ACAT software citation experiment

The intent of this paper is to follow up on a plenary talk given at ACAT 2017 about an experiment in which the conference organizers encourage conference speakers to cite the software they talk about in their proceedings papers. In addition to this talk, the proceedings instructions also made this request. Of course, the goal of this experiment is to move the ACAT community towards citing software, in order to give the authors of the software credit when it is used.

7. Conclusions

Software is essential for the bulk of research today as researchers tell us directly and via their papers. However, it is not cited in papers nearly as often as it is used, leading to developers not getting sufficient credit for their work. In order to change this, the FORCE11 Software Citation working group developed a set of software citation principles that permit software to be published and then cited in papers as other research is. This paper has described the principles, and has provided some examples of how physics software can be cited. This is intended to encourage ACAT authors to cite the software they use in the papers they write in these proceedings. In the future, we will be able to examine these proceedings and see what has worked and what needs to be improved, as well as to examine proceedings from future years to track changes.

References

[1] Hettrick S 2014 It's impossible to conduct research without software, say 7 out of 10 UK researchers URL http://bit.ly/2B8y6Iz

[2] Hettrick S, Antonioletti M, Carr L, Chue Hong N, Crouch S, De Roure D, Emsley I, Goble C, Hay A, Inupakutika D, Jackson M, Nenadic A, Parkinson T, Parsons M I, Pawlik A, Peru G, Proeme A, Robinson J and Sufi S 2014 UK research software survey 2014 URL https://doi.org/10.5281/zenodo.14809

[3] Nangia U and Katz D S 2017 *figshare* URL https://doi.org/10.6084/m9.figshare.5328442.v3

[4] Nangia U and Katz D S 2017 Survey of National Postdoctoral Association - Dataset URL https://doi.org/10.5281/zenodo.843607

[5] Howison J and Bullard J 2016 *Journal of the Association for Information Science and Technology* **67** 2137–2155 ISSN 2330-1643 URL https://doi.org/10.1002/asi.23538

[6] Nangia U and Katz D S 2017 Understanding Software in Research: Initial Results from Examining Nature and a Call for Collaboration *Proceedings of the 13th IEEE International Conference on eScience (eScience 2017)* URL https://doi.org/10.1109/eScience.2017.78

[7] Smith A M, Katz D S, Niemeyer K E and FORCE11 Software Citation Working Group 2016 *PeerJ Computer Science* **2** e86 URL https://doi.org/10.7717/peerj-cs.86

[8] Data Citation Synthesis Group 2014 Joint Declaration of Data Citation Principles Martone, M. (ed), FORCE11, San Diego, CA URL https://doi.org/10.25490/a97f-egyk