Software Papers: improving the reusability and sustainability of scientific software

Neil Chue Hong
Software Sustainability Institute
JCMB, Mayfield Road
Edinburgh, EH9 3JZ, UK
N.ChueHong@software.ac.uk

Brian Hole
Ubiquity Press
Gordon House, Windmill Street
London, W1T 2JB, UK
brian.hole@ubiquitypress.com

Samuel Moore
Ubiquity Press
Gordon House, Windmill Street
London, W1T 2JB, UK
samuel.moore@ubiquitypress.com

ABSTRACT
In this paper, we describe the Journal of Open Research Software, a software metajournal which features peer reviewed software papers describing research software with high reuse potential. We posit that the use of software papers improves the sustainability of scientific software by making them discoverable and citable, as well as asking them questions about the metadata required to use and reuse the software easily.

Keywords
Software papers, metajournals, software sustainability, software reusability, bibliometrics.

1. INTRODUCTION
Until there is a radical change in the way that academic credit is given, the principal record of scientific research is still the peer-reviewed publication. Given that software is a fundamental part of doing science in the digital age, the question we are often asked is: where can someone publish papers which are primarily focused on their scientific software?

There are many reasons for wanting to do this:

- as a record of a particular research object;
- to advertise the work that has been done;
- to allow scrutiny of your work;
- so that other can reproduce your methods;
- to enable reuse amongst other in your research community;
- to build on your work to look at new kinds of studies;
- to allow its reuse for other purposes such as teaching, journalism and citizen science;
- to describe your software such that it can be preserved;
- to enable recognition and reward of your work.

More generally, publishing software itself is considered important for good science [6][7][8] and so enabling transparency for scientific software is also important. There are a number of traditional journals [3] that allow submissions that are primarily about the software, and not necessarily on new algorithms or new science.

One of these journals is the Journal of Open Research Software (JORS) which publishes peer reviewed software papers describing research software with high reuse potential. JORS publishes software papers, which do not contain research results but rather a concise description of scientific software, and where to find it. Papers are only accepted for software which authors agree to make freely available in a public repository. This means that they have been deposited in a digital repository under an open license (such as an Open Source Initiative approved license or public domain Creative Commons Zero license), and are therefore freely available to anyone with an internet connection, anywhere in the world. The software paper itself is licensed under a Creative Commons Attribution (CC-BY) license.

The concept of a software paper is a publication that is designed to make other researchers aware of software that is of potential use to them. In this respect, a software paper can be considered a "metapaper" i.e. principally a paper recording metadata about the software. It describes what problem the software addresses, how it was implemented and architected, where it is stored, and its reuse potential. It is important to note that a software paper does not replace a research article, but rather complements it. When mentioning the software behind a study, a research paper should reference the software paper for further details. The software paper similarly should contain references to any research papers associated with the software. This also enables the software paper to be published before the research papers, if this is appropriate.

JORS submissions consist of a title, abstract and keywords; overview of the software; details of the implementation, architecture and quality control; requirements and dependencies; repository, contributor and license details; and a section on reuse potential. At the time of submission, the software described must be available from a digital repository or source code repository which is suitable for the type of software involved, sustainable (i.e. it must have funding and plans in place to ensure the long-term preservation of the data), allows open licenses and provides persistent identifiers.

The remainder of this position paper considers the challenges of peer reviewing software, the benefits of citation and cross-referencing software, and what this means for the sustainability of scientific software.

2. PEER REVIEW OF SOFTWARE

2.1 Challenges of peer review: judging acceptability vs ensuring quality
A particular challenge for software papers is the concept of peer review of software. Even after 300 years of peer review, the debate continues as to whether peer review gets the balance between judging novelty and acceptability versus ensuring validity and quality (as expressed by Richard Horton's exhortation that "peer review was any more than a crude means of discovering the acceptability — not the validity — of a new finding" [5]) for traditional outputs such as papers. When we consider peer review of scientific software, we must ask whether we can decouple these two sides of scrutiny to make it possible to accomplish the review

in a reasonable length of time. It also questions what we are aiming to achieve by peer reviewing software.

Two general principles guide the reviewing process at the Journal of Open Research Software:

1. We are reviewing the accuracy and quality of the metadata rather than the software, however there will be a minimum level of quality of software required so that it is possible to review.
2. We expect all metapapers to be able to pass after revisions, unless the software is not openly available and/or extremely difficult to reuse.

This means that the Journal of Open Research Software is not directly concerned with novelty, but is concerned about the quality of the metadata. This has some benefits for the objectivity of the review. For instance, it is easy to ask reviewers to check whether the software has an approved license, and whether that license is correctly displayed. Other information that can be checked in this way include checking the persistent identifier to the software and whether sample input and output data is provided. However most of the review is still somewhat subjective: e.g. do the keywords give enough information for a reader to search for the software.

2.2 JORS Review Criteria

Paper contents

a. Is the title of the paper descriptive and objective?
b. Does the Abstract give an indication of the software's functionality, and where it would be used?
c. Do the keywords enable a reader to search for the software?
d. Does the Introduction give enough background information to understand the context of the software's development and use?
e. Does the Implementation and Architecture section give enough information to get an idea of how the software is designed, and any constraints that may be placed on its use?
f. Does the Quality Control section adequately explain how the software results can be trusted?
g. Does the Reuse section provide concrete and useful suggestions for reuse of the software, for instance: other potential applications, ways of extending or modifying the software, integration with other software?
h. Are figures and diagrams used to enhance the description? Are they clear and meaningful?
i. Do you believe that another researcher could take the software and use it, or take the software and build on it?

Deposited software

a. Is the software in a suitable repository?
b. Does the software have a suitable open licence?
c. If the Archive section is filled out, is the link in the form of a persistent identifier, e.g. a DOI? Can you download the software from this link?
d. If the Code Repository section is filled out, does the identifier link to the appropriate place to download the source code? Can you download the source code from this link?
e. Is the software license included in the software in the repository? Is it included in the source code?
f. Is sample input and output data provided with the software?
g. Is the code adequately documented? Can a reader understand how to build/deploy/install/run the software, and identify whether the software is operating as expected?
h. Does the software run on the systems specified? (if you do not have access to a system with the prerequisite requirements, let us know).
i. Is it obvious what the support mechanisms for the software are?

2.3 Further thoughts on reviewing scientific software

Others have also considered what peer review of software might look like, in particular Carl Boettiger. He notes that he is reviewing more and more "software papers", particularly R packages, from authors trying to hack the publication recognition system. As such he comes up with the following ethos for a reviewer [1]:

"I expect the paper to provide the journal's audience with a clear motivation for why the package is useful, and have at least one functioning "wow" example that I can run (by copy-paste) and understand without difficulty (e.g. without referring to code comments or the package manual to understand the function calls and their arguments)."

This effectively asks for the transfer of effort from reader to author: it is up to the author to ensure that the reader can recognise the basic usefulness of the software. Taken to the logical extreme, this would ensure that the software came with full unit and system testing, preferably including checking against some ground truth such as an experimental result. More importantly, this shift of emphasis to the author doing more work before submission to document their software improves the maintainability and sustainability of the code. If it is easier for a general reader to understand how the code functions, it is also easier for an interested reader who intends to reuse or extend the software. Likewise, by requiring software papers to include a permanent identifier such as a digital object identifier (DOI) or permalink to the version in the repository, it is aiding preservation and sustainability of the software by forcing authors to consider what makes a good repository.

A final possibility is to use techniques from software engineering to assess the "quality" of the software objectively, such as cyclometric complexity. Here we must consider the usefulness of these metrics: low cyclometric complexity probably indicates that the code will be easier to understand and change in the future, but do not indicate whether a piece of software will be reusable outside of its original scientific domain. A balance of objective and subjective reviewing will always be required for scientific software.

3. CITATION OF SOFTWARE

A significant feature of software papers is the ability to use the existing publication citation and referencing systems to cite, search, discover and cross-reference. By assigning an identifier to both the software (in the repository) and the software paper (via a DOI), the two are linked such that it becomes much easier to track the usage of the software as it is now a full-blown citable object placed in the references of a research publication, rather than just a footnote to a URL.

With the Journal of Open Research Software we expect there to be an identifier for the deposited software in the digital

repository or source code repository, we issue a DOI for the software paper, and we expect authors to cite their own software paper in research papers which are based on their work. Importantly, as well as the DOI, the software now has a "traditional format" citation which is important as some common tools such as ISI Web of Science and Scopus appear to strip DOIs and URLs out of citations as they import references into their database [9]. DOIs and identifiers also make it easier to use Alt-Metrics frameworks: JORS embeds information from ImpactStory[1] to show the audience engagement by a number of alternative metrics such as Twitter in comparison to all articles indexed in Web of Science that year. This helps authors understand their articles reach, and along with standard article metrics such as views and downloads gives a sense of the interest in their software. The citations of the software paper itself represents a direct measure of the usage of the code.

The "traditional" way of writing a paper to recognize some software would require the generation of new science to put in the paper along with the description of the software used to generate the results This decoupling of the software from the science results means that each can be published and given an identifier at a more suitable point. When the first version of a piece of scientific software is made available, a software metapaper can be written and published immediately. Once the scientific experiments utilising the software are complete they can be published in a research paper which cites the software paper. The software paper can in turn be updated to reference the research paper or to indicate a new version of the software has been released. The ability of DOIs to "point" forwards and backwards allows a much richer cross-referencing.

Of course the reason why we publish is to gain recognition and credit within the academic community. The proof of the effectiveness of journals like JORS will be if they are encouraging the citation of software, and being accepted as evidence for promotion committees. A final benefit might be identified in a change of the citation usage. As Luke Harmon comments [2] the difference is *"Another cool method also exists (Weasel et al. 2008)." versus "We used Weasel et al. (2008) to do amazing things"* – i.e. software papers could encourage the reuse rather than reinvention of software.

4. SUSTAINABILITY OF SOFTWARE

Ultimately, the sustainability of scientific software is based around a few key qualities [4]:

- **Community:** There is a community infrastructure with a common investment *(required for sustainability)*
- **Open:** Software has permissive license *(required for modification)*
- **Defined:** Accurate metadata defines the software and its functionality, dependencies and constraints *(required for preservation)*
- **Extensible:** The software is usable, modifiable for different data, pipelines, purposes *(required for reproducibility)*
- **Runnable:** The software is available and provides the information to operate it *(required for publication)*

To these we would add two more general points:

- **Discoverable:** can I find it?
- **Reusable:** do I know if it useful for me?

We believe that software papers are a pragmatic way of ensuring scientific software meets these criteria, by making other researchers aware of software that might be useful to them. They do this in ways that encourage openness, extensibility and community, whilst checking that it is well defined and runnable. By enforcing the use of suitable repositories, longer term access to and preservation of the software is improved. We therefore posit that software papers are a useful mechanism for improving the reusability and sustainability of scientific software and will be continuing to assess the impact of papers published in the Journal of Open Research Software[2].

5. ACKNOWLEDGMENTS

Our thanks to our colleagues at the Software Sustainability Institute and Ubiquity Press, as well as all the JORS reviewers for their support of the journal. The Software Sustainability Institute is supported by EPSRC grant EP/H043160/1.

6. REFERENCES

[1] Boettiger, C. 2013. *What I look for in 'Software Papers'*. Accessed on 6th September from: http://carlboettiger.info/2013/06/13/what-I-look-for-in-software-papers.html

[2] Boettiger, C. 2013. *Reviewing Software Revisited*. Accessed on 6th September from: http://carlboettiger.info/2013/07/09/reviewing-software-revisited.html

[3] Chue Hong, N. *Which journals should I publish my software in?* Software Sustainability Institute Guide. Accessed on 6th September 2013 from: http://www.software.ac.uk/resources/guides/which-journals-should-i-publish-my-software

[4] Chue Hong, N. 2013. *The Five Stars of Research Software* Accessed on 6th September 2013 from: http://www.software.ac.uk/blog/2013-04-09-five-stars-research-software

[5] Horton, R. *Genetically modified food: consternation, confusion and crack-up*. Med J Aust 2000; 172 (4): 148-149. PMID: 10772580.

[6] Hsu, J. 2012. *Secret Computer Code Threatens Science*. Scientific American. Accessed on 6th September 2013 from: http://www.scientificamerican.com/article.cfm?id=secret-computer-code-threatens-science

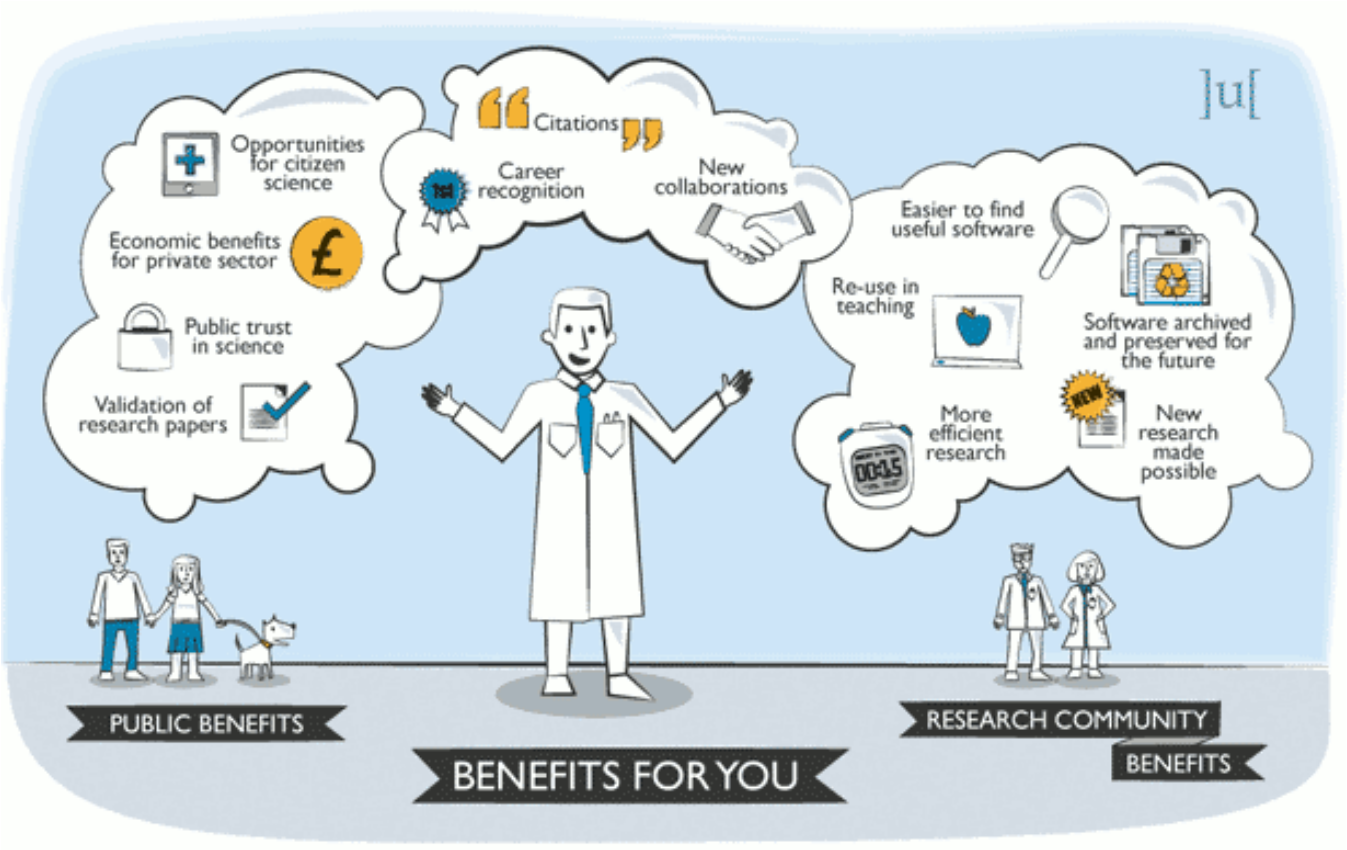
[7] Ince, D.C., Hatton, L., and Graham-Cumming, J. 2012. *The case for open computer programs.* Nature 482, 485–488. doi: 10.1038/nature10836.

[8] Morin, A., Urban, J., Adams, P.D., Foster, I., Sali, A., Baker, D., and Sliz, P. 2012. *Shining Light Into Black Boxes*. Science. Vol. 336 no. 6078 pp. 159-160. DOI: 10.1126/science.1218263.

[9] Piwowar, H. 2010. *Tracking Dataset Citations Using Common Citation Tracking Tools Doesn't Work*. Accessed on 6th September 2013 from: http://researchremix.wordpress.com/2010/11/09/tracking-dataset-citations-using-common-citation-tracking-tools-doesnt-work/

[1] ImpactStory: http://impactstory.org/

[2] JORS: http://openresearchsoftware.metajnl.com/


Opportunities for citizen science
Economic benefits for private sector
Public trust in science
Validation of research papers
Citations
Career recognition
New collaborations
Easier to find useful software
Re-use in teaching
Software archived and preserved for the future
More efficient research
New research made possible
PUBLIC BENEFITS
BENEFITS FOR YOU
RESEARCH COMMUNITY BENEFITS