ZBW *Publikationsarchiv*

Publikationen von Beschäftigten der ZBW – Leibniz-Informationszentrum Wirtschaft
Publications by ZBW – Leibniz Information Centre for Economics staff members

Borst, Timo

Conference Paper

Sustainable Software as a Building Block for Open Science

Suggested Citation: Borst, Timo (2015) : Sustainable Software as a Building Block for Open Science, In: Schmidt, Birgit Dobreva, Milena (Ed.): New Avenues for Electronic Publishing in the Age of Infinite Collections and Citizen Science: Scale, Openness and Trust. Proceedings of the 19th International Conference on Electronic Publishing, ISBN 978-1-61499-562-3, IOS Press, Amsterdam, pp. 31-36,
https://doi.org/10.3233/978-1-61499-562-3-31

This Version is available at:
http://hdl.handle.net/11108/236

Kontakt/Contact
ZBW – Leibniz-Informationszentrum Wirtschaft/Leibniz Information Centre for Economics
Düsternbrooker Weg 120
24105 Kiel (Germany)
E-Mail: info@zbw.eu
https://www.zbw.eu/de/ueber-uns/profil-der-zbw/veroeffentlichungen-zbw

Standard-Nutzungsbedingungen:
Dieses Dokument darf zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden. Sie dürfen dieses Dokument nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, aufführen, vertreiben oder anderweitig nutzen. Sofern für das Dokument eine Open-Content-Lizenz verwendet wurde, so gelten abweichend von diesen Nutzungsbedingungen die in der Lizenz gewährten Nutzungsrechte.

Terms of use:
This document may be saved and copied for your personal and scholarly purposes. You are not to copy it for public or commercial purposes, to exhibit the document in public, to perform, distribute or otherwise use the document in public. If the document is made available under a Creative Commons Licence you may exercise further usage rights as specified in the licence.

http://creativecommons.org/licenses/by-nc/3.0/deed.en_US

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Mitglied der Leibniz-Gemeinschaft

New Avenues for Electronic Publishing in the Age of Infinite Collections and Citizen Science
B. Schmidt and M. Dobreva (Eds.)
© 2015 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-562-3-31

Sustainable Software as a Building Block for Open Science

Timo BORST[1]
ZBW – Leibniz Information Centre for Economics

Abstract. In the context of Open Science, almost every 'traditional' research activity and output has been affected and transformed by means of web based technology. New forms of research output have emerged, among them software as an important means and method for data driven science. But how can software be treated as scholarly work, and how can it be integrated into a digital research infrastructure? The paper depicts software development related to Open Science and points out some future directions for software to become part of a sustainable research infrastructure.

Keywords. Research software, research infrastructures, Open Science

1. Software as a Factor for Open Science

In recent years, the digital transformation of the system of scholarly communication has often been recognized as one of the basic outcomes and challenges in modern information society. Almost any stage of the scientific process has been affected, be it workflows like the generation, distribution and sharing of scientific results, or their reviewing and communicating. Central, if not any of the scientific processes happen by means of digital environments including tools and applications, while core processes like dissemination and communication of scientific results preferably take place in web based environments.

At the same time, the process of software development including producing, distributing, sharing and modifying program code has undergone a similar change, which at first sight does not look too surprising. Where else than in computer science and industry would you expect first a change or shift in the use of digital environments? Have the first emails not been sent by computer experts, the first user groups communicating via mailing lists not been established by those experts, and the sharing, modifying and enhancing of e.g. LINUX been conducted by them? Where in general the answer may look obvious, in the scientific realm it becomes a bit more tricky: The systematic distribution and sharing of software as both a basis for and outcome of a digital system of scholarly communication has neither been recognized so far, nor explicitly been put on the agenda like comparable workflows in the scientific publishing process. Rather, it may be still conceived as in an early stage of incubation.

However, there are indicators that software development for research purposes has already adapted itself to requirements or symptoms of Open Science. One striking

[1] ZBW – National Library of Economics, Düsternbrooker Weg 120, 24105 Kiel, Germany; E-mail: t.borst@zbw.eu.

observation may be that in the context of research data it has been demanded to publish not only the data itself, but also the program code to generate or manipulate it [1]. Publishing under the conditions of Open Access, and publishing Open Source Software converge from a both conceptual and operational point of view, especially when it comes to the reproducibility and replicability of scientific results by means of program code. Moreover, the following trends may indicate the importance of sustainable software as both an enabler and a result of Open Science [2]:

Under the general label of 'openness', we can observe a convergence of the three movements Open Science, Open Data and Open Source. Apart from the more recent endeavors in managing and provisioning research data and algorithms to calculate them, namely the third topic has its roots in a quite early practice independent from the other two, but with a certain impact on them. A significant difference still can be seen in the openness of Open Source Software towards commercial purposes, so these purposes have become an important driver.

With the emergence of social networks, graph based approaches towards the modelling of relations and collaborations between (social) entities have become very prominent and successful. But long before that, concepts and tools for graph-based version management have been introduced into distributed software development with an explicit history of software releases. Version management tools like Apache Subversion (SVN) or Concurrent Version System (CVS) arouse before or parallel to the World Wide Web (WWW), while web-based platforms like Sourceforge [3] and particularly GitHub [4] have fostered the distribution, tracking, monitoring and reuse of software in terms of awareness and collaboration. Regarding the variety of uses of GitHub as a popular distribution platform for code, data and text in sciences [5], one may seriously take these adaptations into account as steps towards an operational Open Science. For instance, it has been suggested to adapt traditional bibliographical metrics and to measure the impact of software distributors by means of some kind of 'page rank algorithm' to calculate the most cited (=forked) git repositories [6].

In analogy to 'Citizen Science' as a synonym for an open and collaborative science activity, one may regard the character of a 'Citizen Developer' contributing code in an open environment for like-minded persons. The basic idea is that in a role as a 'citizen developer', one may still contribute to Open Software projects without an institutional or professional background. In contrast and in a perhaps more sophisticated enhancement of the role as a 'citizen scientist' – where participants are mostly committed to mass data contribution –, the 'citizen developer' acts in a both collaborative and individualistic environment, leaving his or her digital footprints in software repositories being part of a global software environment for science and research.

Now, what to infer from these observations? If software is becoming more and more constitutive and public as crucial contribution to Open Science, it will be essential to provide an environment for managing this work similar to 'traditional' research output.

2. Upcoming Challenges: Research Software vs. Infrastructure Software from a Stakeholder's Perspective

From the point of view of infrastructure providers, software developed by scientists may be called 'immature' in the sense, that its origin is primarily individual, local and temporal. Being absorbed by a specific research question, a researcher normally will not care very much about the engineering aspects of his or her software, nor will he or she have the time to 'harden' the code for potential reuse. Even for professional, full-time committed software developers this requirement is still bothersome and definitely not first-order activity. On the other hand, research software built by scientists is a constitutive part of a future environment for Open Science, in the sense that it will be needed for later reuse, validation and reproduction of scientific results in connection with other scientific outcomes, e.g. research data. Hence, the crucial question can be put as how to integrate (individual) research software with something like 'sustainable software infrastructure', so it can finally become part of the latter?

3. Future Directions and Recommendations for Sustainable Research Software

In the following, we recommend some principles and steps to be taken, which are obviously adopted from existing workflows in order to align the development of research software with other research and publishing activities. The principles are formulated as requirements towards the authors resp. the publishers of research software from the point of view of research infrastructure providers. They may be in line with requirements of other stakeholders like research funders, but this has to be negotiated yet.

3.1. Software Code as Open Source

As already stated, Open Science relates to Open Source Software in an intuitive, but not yet operative way. For the purpose of the transition from idiosyncratic code to reusable and adaptable, quality assured packages as part of a common software infrastructure, it is essential to fully provide transparent source code plus the software environment (libraries, runtime environment, virtual machines, container etc.) to run that code. Consider, e.g. the calculation of a regression analysis on the basis of a self-developed algorithm despite the fact that there are already reliable packages e.g. at CRAN [7]: in case of inconsistencies, a reviewer or adopter of the results would not be able to distinguish between wrong data or wrong code resp. algorithms. To be 'open' is not just an attitude or style adopted from another context, but a *conditio sine qua non* for reproducing, reviewing and revising research results based on software.

Publishing software as Open Source implies some legal, organizational and technical aspects which must be cleared in advance, best on an institutional level. In contrast to other research output like publications or data, a reuse of software for commercial purposes might be more likely – but there are a couple of proven Open Source licenses suitable for regulating these concerns.

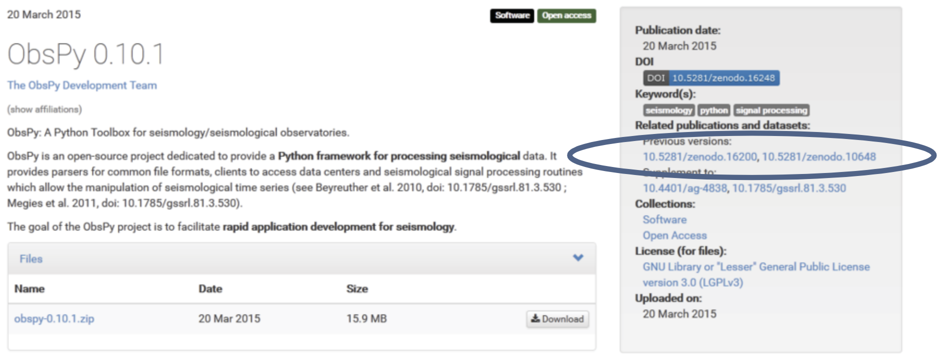
3.2. Publishing and Sharing of Code in Conjunction with Data and Additional Material

It is good practice to publish all material related to a scientific work if not simultaneously, but in one location – at least with references to their physical location –, so the user gets a quick overview on and access to the total scholarly work and its components. A Digital Object Identifier (DOI) may be useful to resolve to the overall work ('jump page'), but can also be attached to its constituents (publication, data, code, blog posts, slides, tweets …).

For publishing and sharing of code, GitHub has become a trendy and widely used platform for 'social coding' [8]. Adopting common vocabulary and comprehensive workflow steps partly derived from other code versioning systems ('commit', 'fork', 'branch'), plus the typical 'social media' characteristics of global visibility, tracking and awareness have lifted GitHub to the most popular and widely used platform for code management and sharing, forcing e.g. Google to shut down their corresponding service and users to migrate their code repositories to GitHub [9]. However, several objections have been made towards GitHub as a service for global code management: (a) As an internet service, the platform is exposed to hacking, (b) the provider – a start-up company from San Francisco – is heavily dependent from external venture capital and potential market interests, and (c) the cloning of a GitHub repository is no real substitute for a decentralized, 'officially' redundant backup infrastructure [10]. Hence, GitHub may be rather regarded as a model for publishing and sharing research software, not as the only or primary host for publicly financed research output. It serves as a platform for disseminating code, but should be backed up by local code git repositories.

3.3. Making Code Citable

Although this aspect may be conceived as an integral part of publishing code, it deserves special attention and handling: So far, the citing of software code similar to traditional research output has not been put explicitly on the agenda of stakeholders like publishers or research funders, nor does it yet belong to the 'impact story' of researchers. On the other hand, citing is already supported by platforms like Figshare [11] or Zenodo [12], both of them integrating GitHub repositories as the original platform for code publishing. In Zenodo, each version of software can be referenced by its own DOI (cf. Figure 1), hence associated with the research supplement generated with that version (data, publication). This is especially important in the case of software packages which include non-standard algorithms, and where the results from the calculation are dependent on the distributor or even the version of software.



20 March 2015

Software Open access

ObsPy 0.10.1

The ObsPy Development Team

(show affiliations)

ObsPy: A Python Toolbox for seismology/seismological observatories.

ObsPy is an open-source project dedicated to provide a **Python framework for processing seismological** data. It provides parsers for common file formats, clients to access data centers and seismological signal processing routines which allow the manipulation of seismological time series (see Beyreuther et al. 2010, doi: 10.1785/gssrl.81.3.530 ; Megies et al. 2011, doi: 10.1785/gssrl.81.3.530).

The goal of the ObsPy project is to facilitate **rapid application development for seismology**.

Files

Name	Date	Size	
obspy-0.10.1.zip	20 Mar 2015	15.9 MB	Download

Publication date:
20 March 2015
DOI
DOI 10.5281/zenodo.16248
Keyword(s):
seismology python signal processing
Related publications and datasets:
Previous versions:
10.5281/zenodo.16200, 10.5281/zenodo.10648
Supplement to:
10.4401/ag-4838, 10.1785/gssrl.81.3.530
Collections:
Software
Open Access
License (for files):
GNU Library or "Lesser" General Public License version 3.0 (LGPLv3)
Uploaded on:
20 March 2015

Figure 1. Screenshot from a Zenodo record.

Solutions like the one from Zenodo may be a good way to publish software by means of more 'official' platforms, while at the same time pointing to continuous development and deployment on platforms like GitHub.

3.4. Research Representation, Analysis and Evaluation

To become more visible as genuine research output, references to software code should become part of the scholarly record implemented by researcher identifier systems like ORCID [13], VIAF [14] or – on a more national level – DAI [15] and GND [16]. For instance, in ORCID the creation of an XML instance for describing a piece of published code should follow the XML scheme for describing a publication by using e.g. the APA style [17]:

```
<orcid-work>
...
<work-citation>
       <work-citation-type>formatted-apa</work-citation-type>
       <citation>
       Gregory Jefferis, James Manton, & Ben Sutcliffe. (2015).
       nat: nat 1.6.5. Zenodo.
       http://doi.org/10.5281/zenodo.17558
       </citation>
...
</orcid-work>
```

4. Conclusions

Considering software as explicit research output is still in its very beginning, and so is the integration into existing research infrastructures. Software development is not a primary research activity, but becoming more crucial especially in data driven sciences. From the point of view of a research infrastructure, the formal basics for identifying, citing and integrating software as research output are already there – what we now do need is more investigation on the curation, maintenance and preservation of this software, so it can become integral part of future research.

References

[1] http://openeconomics.net/principles/ (accessed 2 June 2015).

[2] M. D. Hanwell et al., Sustainable Software Ecosystems for Open Science: 15 Years of Practice and Experience at Kitware, *arXiv:1309.2966v1* (2013), DOI: 10.6084/m9.figshare.790756

[3] http://sourceforge.net/ (accessed 2 June 2015).

[4] https://github.com/ (accessed 2 June 2015).

[5] P. Krill, GitHub rolls out the red carpet for scientists (2014), http://www.javaworld.com/article/2157321/open-source-tools/github-rolls-out-the-red-carpet-for-scientists.html (accessed 2 June 2015).

[6] F. Thung et al., Network Structure of Social Coding in GitHub, 17th *European Conference on Software Maintenance and Reengineering (CSMR) 2013* (2013), 323-326. DOI: 10.1109/CSMR.2013.41

[7] http://cran.r-project.org/ (accessed 2 June 2015).

[8] L. Dabbish et al., Social coding in GitHub: transparency and collaboration in an open software repository, *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work* (2012), 1277-1286. DOI: 10.1145/2145204.2145396

[9] http://google-opensource.blogspot.jp/2015/03/farewell-to-google-code.html (accessed 2 June 2015).
[10] http://blog.printf.net/articles/2015/05/29/announcing-gittorrent-a-decentralized-github/ (accessed 2 June 2015).
[11] http://figshare.com/ (accessed 2 June 2015).
[12] https://zenodo.org/ (accessed 2 June 2015).
[13] http://orcid.org/ (accessed 2 June 2015).
[14] http://viaf.org/ (accessed 2 June 2015).
[15] https://www.surf.nl/en/themes/research/research-information/digital-author-identifier-dai/digital-author-identifier-dai.html (accessed 2 June 2015).
[16] http://www.dnb.de/DE/Standardisierung/GND/gnd_node.html (accessed 2 June 2015).
[17] http://blog.apastyle.org/apastyle/2015/01/how-to-cite-software-in-apa-style.html (accessed 2 June 2015).