Publish Your Software: Introducing the *Journal of Open Source Software (JOSS)*

Daniel S. Katz
National Center for Supercomputing Applications; University of Illinois, Urbana–Champaign

Kyle E. Niemeyer
Oregon State University

Arfon M. Smith
Space Telescope Science Institute

Editors:
Jeffrey Carver, carver@cs.ua.edu; Damian Rouson, damian@sourceryinstitute.org

Software is essential to most research today. We know this because we have asked researchers and examined their outputs. A 2014 study of UK Russell Group Universities[1] reports that approximately 90 percent of academics surveyed said that they use software in their research, and more than 70 percent said that their research would be impractical without it. About half of these UK academics reported that they develop their own software while in the course of doing research. Similarly, a 2017 survey of US National Postdoctoral Association members found that 95 percent used research software, and 63 percent said that their research would be impractical without it.[2] An initial study of software's role in journal articles examined three months of the journal *Nature* and found that during this period, of the 40 research articles *Nature* published, 32 mentioned software and averaged 6.5 software package mentions per article.[3]

However, even though software is a critical part of modern research, its publication, acknowledgement, and citation are not well-supported across the scholarly ecosystem.[4] Academic publishing has not changed substantially since its inception. Science, engineering, and many other academic fields still view research articles as the key indicator of research productivity, with research grants being another important indicator. Yet, the research article is inadequate to fully describe modern, data-intensive, computational research.

The *Journal of Open Source Software* (*JOSS*; joss.theoj.org)[5] focuses on research software and its place in the scholarly publishing ecosystem. Its goal is to make it easy for authors to publish a paper about their software, mostly focused on the software itself, and then be credited when this software is used based on the users citing the *JOSS* paper. Thus, the journal's selling point, as stated in the author guidelines, is as follows:

> *If you've already licensed your code and have good documentation, then we expect that it should take less than an hour to prepare and submit your paper to* JOSS.

Copublished by the IEEE CS and the AIP
1521-9615/18/$33 ©2018 IEEE

JOSS is

- **Built on GitHub.** We use GitHub so that we can take advantage of numerous of its features (for example, the use of issues for reviews; rapid interaction between author, reviewer, and editor; and notifications) and authors' and reviewers' general familiarity with GitHub.
- **Open access.** Papers submitted to and accepted by *JOSS* are open access (CC-BY licensed), and authors retain the copyright to their work. The software components of the publications are licensed under an Open Source Initiative (OSI) approved license.
- **Intended for research software.** *JOSS* attempts to be general to all research software, not just for science or any single field. The main question we ask of submitters is, "Is it likely that users of this software will want to cite it?"
- **Intended to be easy to use for authors.** Authors with well-documented code in a public code repository simply need to create a directory in that repository that contains a short paper in Markdown format; in that paper, they then declare the authors of the software, the purpose of the software, and any needed references.
- **Intended to be easy to use for reviewers.** Reviewers work through a checklist of items related to *JOSS*'s conflicts of interest policy, the code and documentation in the software repository, and the software paper itself, as further discussed in the next section.
- **Supported by NumFOCUS.** NumFOCUS (numfocus.org) is a 501(c)(3) nonprofit that supports and promotes world-class, innovative, open source scientific computing. It provides a forum for such projects to work through issues, and provides a mechanism for these projects to accept funding. Open Journals (http://theoj.org) is a collection of open source, open access journals, of which *JOSS* is the flagship publication. Other journals based on the same model are in varying stages of spinning up.
- **Endorsed by and affiliated with OSI.** The OSI (opensource.org) is a global nonprofit organization that protects and promotes open source software, development, and communities by championing software freedom in society through education, collaboration, and infrastructure, stewarding the Open Source Definition (OSD) and preventing abuse of the ideals and ethos inherent to the open source movement. In support of this mission, *JOSS* requires that the software it publishes uses an OSI-approved license.
- **Indexed.** *JOSS* is listed on Sherpa/Romeo, and *JOSS* papers are indexed by ADS (http://adsabs.harvard.edu), which in turn is indexed by Google Scholar.
- **Connected to other code reviewing systems.** For submissions of software that has already been reviewed under rOpenSci's rigorous onboarding guidelines,[6] *JOSS* does not perform further review; the editor in chief fast-tracks such submissions to acceptance. And this could be extended to work with other trustworthy systems.

JOSS PROCESS

The typical *JOSS* submission and review process (shown in Figure 1) follows the steps described below.

1. An author submits an article, including a link to software, to *JOSS* using the web application and submission tool. The article is a Markdown file named paper.md, visibly located in the software repository (in many cases, placed together with auxiliary files in a paper directory).

2. Following a routine check by a *JOSS* administrator, a "pre-review" issue is created in the joss-reviews (https://github.com/openjournals/joss-reviews) GitHub repository. In this pre-review issue, an editor is assigned who then identifies and assigns a suitable reviewer. The editor then asks the automated bot Whedon to create the main submission review issue.

3. The reviewer conducts the submission review in the issue by working through a checklist of review items:

- agreement with JOSS policies on conflicts of interest and code of conduct;
- general checks to ensure that the source code is available, an OSI-approved license was used, the software version in the paper and the repository match, and the submitter is an author of the software;

- checks on functionality, including installation and performance claims, if appropriate;
- checks on documentation, which should include a statement of need for the software, installation instructions, example usage, and documentation of functionality, tests, and community guidelines; and
- checks on the paper itself, including ensuring that the author list is reasonable, the statement of need is included, and sufficient and well-structured references are present.

The author, reviewer, and editor discuss any questions that arise during the review, and once the reviewer completes the checks, he or she notifies the submitting author and editor. *JOSS* reviews are discussions—in the open within a GitHub issue—between the reviewer(s), author(s), and editor. Like a true conversation, discussion can go back and forth in minutes or seconds, with all parties contributing at will. This contrasts with traditional journal reviews, where the process is merely an exchange between the reviewer(s) and author(s) via the editor, which can take months for each communication and, in practice, is usually limited to one to three exchanges due to that delay.[7] The reviews are subject to a code of conduct (https://github.com/openjournals/joss/blob/master/CODE_OF_CONDUCT.md); both authors and reviewers must confirm that they have read and will adhere to this code of conduct, during submission and with their review, respectively.

4. After the review is complete, the editor asks the submitting author to make a permanent archive of the software (including any changes made during review) with a service such as Zenodo or Figshare, and to post a link to the archive in the review thread. This link, in the form of a DOI, is attached to the submission.

5. The editor in chief produces a compiled PDF, updates the *JOSS* website, deposits Crossref metadata, and issues a Crossref DOI for the submission.

6. Finally, the editor in chief updates the review issue with the *JOSS* article DOI and closes the review. The submission is then accepted into the journal.

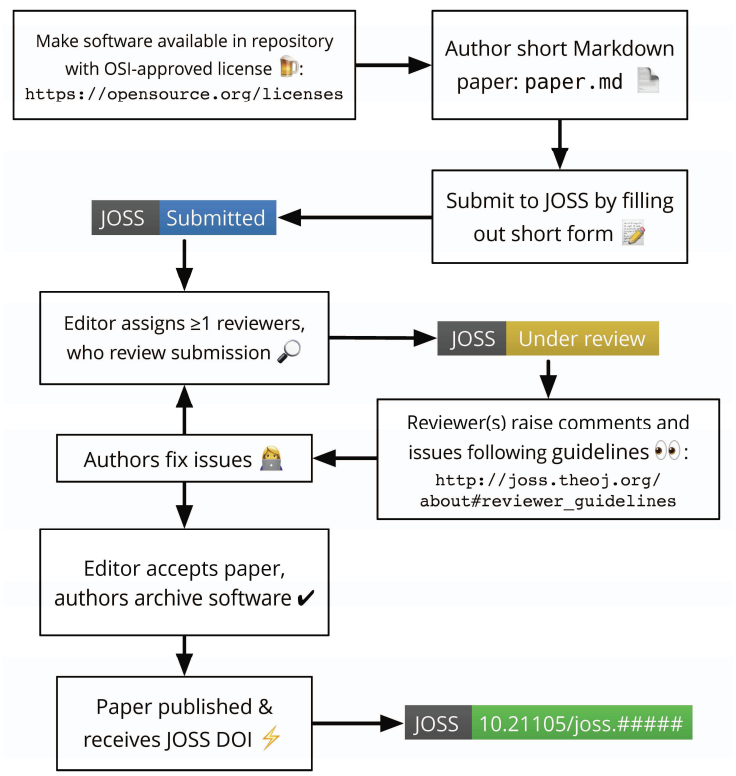

Figure 1. The *JOSS* submission and review flow, including the various status badges that can be embedded on third-party settings such as GitHub README documentation.[8]

STATUS AND FUTURE DIRECTIONS

JOSS received its first submission in May 2016, and as of 22 January 2018, has published 206 papers, with 63 more submitted and being reviewed or otherwise processed. The most-cited articles from *JOSS*'s first 20 months are Daniel Foreman-Mackey's article on corner.py[9] and Conrad Sanderson and Ryan Curtin's article on Armadillo,[10] with 129 and 86 citations, respectively, according to Google Scholar.

The number of submissions and the fact that we have 19 volunteer editors seem to indicate that *JOSS* is satisfying a need. *JOSS* is not the final answer to the problem we ultimately want to solve: authors receiving credit directly for their software. They should not need to write papers about their software to advertise and fit it within the journal system. Nevertheless, our approach is a good intermediate step to highlight the need for recognition and credit for software.

With general awareness of *JOSS* in the community increasing, the *JOSS* editorial team has been contacted by other journals and publishers interested in our approach. These journals generally either want to accept software papers in their journals but desire a deeper review of the software, or want to add software reviews to their editorial process for scientific papers but do not have the expertise in their reviewer pool. Thus, we are considering allowing other journals to request a *JOSS* review as part of their editorial process when software forms a major part of a submission. One publisher interested in such an approach is the American Astronomical Society (AAS), which counts the high-profile *Astrophysical Journal* among its publications.

In addition to potentially collaborating with other journals, we are also in the process of generalizing the *JOSS* infrastructure to allow other journals to be easily created with the same tools. Supported by a small grant from the Alfred P. Sloan Foundation, the core *JOSS* application (https://github.com/openjournals/joss) and the Whedon bot (https://github.com/openjournals/whedon-api) are being adapted to support multiple *JOSS*-like publications. Led by a subset of the *JOSS* editorial team, the *Journal of Open Source Education* will likely be the first new publication to use the generalized infrastructure.

REFERENCES

1. S. Hettrick et al., "UK Research Software Survey 2014," *dataset*, Univ. of Edinburgh on behalf of Software Sustainability Inst., 2015; doi.org/10.7488/ds/253.
2. U. Nangia and D.S. Katz, "Track 1 Paper: Surveying the US National Postdoctoral Association Regarding Software Use and Training in Research," *Proc. Workshop on Sustainable Software for Science: Practice and Experiences* (WSSSPE 5.1), 2017; doi.org/10.6084/m9.figshare.5328442.
3. U. Nangia and D.S. Katz, "Understanding Software in Research: Initial Results from Examining Nature and a Call for Collaboration," *Proc. Workshop on Sustainable Software for Science: Practice and Experiences* (WSSSPE 5.2), 2017; doi.org/10.1109/eScience.2017.78.
4. K.E. Niemeyer, A.M. Smith, and D.S. Katz, "The Challenge and Promise of Software Citation for Credit, Identification, Discovery, and Reuse," *J. Data and Information Quality*, vol. 7, no. 4, 2016, p. 16.
5. A.M. Smith et al., "Journal of Open Source Software (JOSS): Design and First-Year Review," *PeerJ Computer Science*, vol. 4, 2018, p. e7.
6. K. Ram,, N. Ross, and S. Chamberlain, "Lightning Talk: A Model for Peer Review and Onboarding Research Software," *Proc. Fourth Workshop on Sustainable Software for Science: Practice and Experiences* (WSSSPE 4), 2016; ceur-ws.org/Vol-1686/WSSSPE4_paper_13.pdf.
7. J.P. Tennant et al., "A Multi-Disciplinary Perspective on Emergent and Future Innovations in Peer Review [version 2; referees: 2 approved with reservations]," *F1000Research*, vol. 6, no. 1151, 2017; doi.org/10.12688/f1000research.12037.2.
8. K.E. Niemeyer, "JOSS Publication Flowchart," Figshare, 2017; doi.org/10.6084/m9.figshare.5147773.v1.

9. D. Foreman-Mackey, "corner.py: Scatterplot Matrices in Python," *J. Open Source Software*, vol. 1, no. 2, 2016, p. 24.
10. C. Sanderson and R. Curtin, "Armadillo: A Template-based C++ Library for Linear Algebra," *J. Open Source Software*, vol. 1, no. 2, 2016, p. 26.

ABOUT THE AUTHORS

Daniel S. Katz is the assistant director for Scientific Software and Applications at the National Center for Supercomputing Applications, and a research associate professor in the Departments of Computer Science and Electrical and Computer Engineering and the School of Information Sciences at the University of Illinois, Urbana–Champaign. His research interests include applications, algorithms, fault tolerance, programming in parallel and distributed computing, citation and credit mechanisms and practices associated with software and data, organization and community practices for collaboration, and career paths for computing researchers. Katz received a PhD in electrical engineering from Northwestern University. Contact him at d.katz@ieee.org.

Kyle E. Niemeyer is an assistant professor of mechanical engineering in the School of Mechanical, Industrial, and Manufacturing Engineering at Oregon State University. His research interests include the development of advanced numerical methods for modeling of combustion and reactive flows, and computational modeling of multiphysics flows for applications in aerospace, transportation, and energy systems. Niemeyer received a PhD in mechanical engineering from Case Western Reserve University. Contact him at kyle.niemeyer@oregonstate.edu.

Arfon M. Smith is the head of the Data Science Mission Office at the Space Telescope Science Institute. His research interests include academic credit models, crowd-sourcing/citizen science, interstellar medium, open source software, and scholarly publishing. Smith received a PhD in astrochemistry from the University of Nottingham. Contact him at arfon@stsci.edu.