EAI Endorsed Transactions
on Cognitive Communications
Editorial EAI.EU

Software Articles and Open Source Research Software: The Why, the What, and the How

Vincent A. Cicirello*

Computer Science, Stockton University, 101 Vera King Farris Drive, Galloway, NJ 08205

Abstract

In this editorial, I make the case for why you should write and publish software articles about your open source research software. I begin by discussing some of the benefits of releasing your research software as an open source project. Then, I explain what a software article is in general, as well as why you should consider writing one about your research software, such as increased reproducibility, increased software reuse, the potential to increase the impact that your research has on the work of others, the potential to gain new collaborators, among other advantages. Open source enables these things, and software articles enhance the process by increasing discoverability, as well as by placing the software within a broader research context. I then discuss how to proceed to write a software article more specifically for submission to the *EAI Transactions on Cognitive Communications*, using either the "research article" or "short communications" article types.

Received on 21 January 2021; accepted on 22 January 2021; published on 28 January 2021

Keywords: cognitive communications software, impact, open source, reproducible research, reuse, software papers

doi:10.4108/eai.28-1-2021.168507

Why open source research software?

Software is integral to research activities within scientific, engineering, and technological domains. And in this regard, cognitive communications, the theme of this journal, is no different. The roles that software plays within research is varied, and includes modeling and simulation, data analysis, embedded systems, optimization, implementations of domain specific algorithms, and so forth. Research is an activity that is both a producer as well as a consumer of software. If a researcher develops a novel algorithm for problem X, then presumably they will implement their new algorithm to validate the new approach to whatever problem X is—thus, serving as a producer of software. Perhaps validation requires analyzing system performance in simulation. They can implement the simulation environment, or perhaps there is simulation software available that they can instead utilize—thus, research can also be a consumer of software.

Reproducible results: Research articles often include an experimental results section, where the authors describe their experimental assumptions, values for parameters of their technical approach, operating system, programming language, test system details (e.g., CPU speed, memory, etc), among other things. If they do this well, then another researcher in theory can reproduce their experiments with nothing more than the text of the paper as a resource. However, some details may not seem sufficiently important to reproducibility to include in the text of the paper. The solution is simple: release an open source implementation, and link to it from your research article. Others can consult your source code directly for the details that you wouldn't normally write in the text of a paper, and can easily reproduce your results using your code. Note that this model is not what I mean by a software paper. The model described here is a regular research article that includes an implementation among its contributions as an online supplement.

Reusable research components: The opening paragraph alludes to the idea of reusable research components, with the example of using an existing simulation framework to validate a new approach to a research problem rather than building the simulation from scratch. For example, within the scope of this journal, you might need a cognitive radio simulator.

*Co-Editor-in-Chief. Email: vincent.cicirello@stockton.edu. Website: https://www.cicirello.org/

You can implement your own, but you may be more productive to use an existing simulation framework so that you can focus more of your time on your own research questions. For example, Haider *et al.* (2017) evaluates and compares the features of several open source cognitive radio network simulators. Unless your research is focused on innovating new cognitive radio simulators, a more productive use of your time is to utilize one of these or something similar.

Increased research impact: Releasing your research software as open source may increase your impact. If your source code is not openly available, readers of your papers may utilize your ideas, reimplement your approach, and cite your work. But they may be more likely to do so if they are able to directly use your code.

Gaining collaborators: If you develop your research software as an open source project, then you may benefit from the open source community at large. Other researchers using your software may contribute to your project, either in small ways such as reporting bugs or suggesting features, or in more substantial ways such as contributing code enhancements. Whether small or large contributions, there is potential to lead to broader research collaborations—perhaps discovering a common research question to jointly explore.

Why write (and what is) a software article?

Thus far, I outlined a few reasons to release your research software as an open source project. But is it sufficient to host your source code on GitHub, GitLab, BitBucket, or some other cloud host? Well, it might be. But then again, maybe not. In 2018, GitHub reported hosting over 100 million source code repositories (Warner, 2018). Finding relevant software to meet the research needs of your project amidst such a vast collection of open source is a daunting task.

How do you find open source research components to effectively enable *reuse*? And how do others find your open source projects to enable *increased research impact*? More importantly, how do you find adequately vetted open source research software, so that you can feel confident in its quality and suitability for your project?

This is part of the role of a *software article* and there is a growing number of refereed journals devoted to publishing them. Two examples are the *Journal of Open Source Software (JOSS)* (Katz *et al.*, 2018; Smith *et al.*, 2018) and the *Journal of Open Research Software (JORS)* (Hong *et al.*, 2013). Neither one is focused on any one research discipline, and their scope covers all fields where software is important. They publish short papers, much shorter than a traditional research paper, that primarily summarize the research purpose of the software and its primary functionality. Both consider the software to be an integral part of the paper—thus, the very short length of the papers themselves. Indeed, *JORS* even describes such papers as metapapers. The founders of *JOSS* argue that the primary mechanism for assessing academic credit is citations and publishing papers, but that this traditional approach excludes software from the process (Smith *et al.*, 2018). Thus, one of their stated reasons for publishing software papers is enabling software citation, with the paper serving as an "entry point" to the software (Smith *et al.*, 2018).

In addition to the above generalist journals, others focus on software papers within specific fields, such as the *Journal of Statistical Software (JSS)* (`https://www.jstatsoft.org/`). One way that *JSS* differs from the more general software paper journals is that it publishes full research articles, where the primary contribution is novel open source software, and where the paper itself explains underlying concepts, compares other existing implementations and related software, discusses the design principles of the software, etc.

Why write a software paper? You can enable reproducibility without doing so (e.g., link to the source code of experiments from your regular research article). But this alone helps very little with the other reasons for open source research software. The source code for your experiments is probably not ready for easy integration into other projects, or if it is, it may not be obvious how or why to do so. For reproducibility, your experiment source code should strictly run your experiments in the same way that you executed them. Your aim there is not to show how to reuse your code within other systems. You absolutely should consider that approach, supplementing your regular research papers with links to source code that reproduces your experiments. But if you want to maximize the benefits of open source research software (e.g., reusable research components and increased impact), you should consider additionally writing a software article. The software article increases discoverability of your research software since researchers can find it during literature searches. And if written well, they will learn from your software paper the trade-offs associated with using your research software versus others, as well as how and why to integrate your software into their project. This leads to greater reusability, and potential increased impact.

How to write a software article for the *EAI Transactions on Cognitive Communications*?

EAI Transactions on Cognitive Communications is not devoted to software papers, and does not have an article type specific to software papers. However, we have multiple article types that can be appropriate for software papers depending upon the nature and content of your paper.

Scope: First and foremost, your paper and your software that it discusses must be within the scope of the journal, found on the journal's website (https://eudl.eu/journal/cogcom). Among other things, the scope includes artificial intelligence, machine learning, or data analytics for cognitive communications systems; bioinspired communications systems; self-organizing, self-adaptive, and self-aware networks; trust, security, and privacy in cognitive communications; and many other cognitive communications topics (resource allocation, optimization, etc). The full list of topics is online.

Types of software: The type of software that is appropriate for the subject of a software paper is varied, including a library that you developed that includes implementations of algorithms that are useful to cognitive communications systems, a simulation framework, a model implemented within an existing simulation framework, an entire application, etc.

Article type and content: The *EAI Transactions* support several article types (https://eudl.eu/instructions), two of which are relevant for software papers, and the expected content may vary somewhat depending upon the type you choose. The two relevant types are *research article* and *short communications*.

Research article: A software paper submitted as a *research article* will be reviewed as a research article. It must make a novel research contribution. However, the primary contribution can be the software. For example, perhaps the algorithms are not new, but your contribution is a new implementation that packages them together in a way that has the potential to significantly impact the research community. Your article should describe the functionality and make clear the relevance to cognitive communications, and include a discussion of how others may integrate it into their systems, perhaps with an example. Like all research articles, you should discuss any background concepts needed to understand the paper, as well as any related software. In your methods or approach section, you should discuss the design rationale behind your major implementation decisions, and possibly include your system architecture. You should include some form of evaluation, which could be a discussion of your research software within the context of related software packages, such as functionality that your software provides that others do not, and limitations of your software relative to other available packages.

Short communications: Instead, you can write a *short communications* to present your open source research software. Short communications are expected to make novel contributions, but would be a scaled down version of the research article type, likely including the description of the high-level functionality, a discussion of the relevance to cognitive communications, a discussion of related software packages, and a discussion of the potential impact of the software.

Recommendations for the software: We don't have a specific software paper type, so we don't have any strict requirements for the software itself. However, to be most effective (e.g., potential increased reusability and impact, etc), I have several recommendations:

- **OSI-approved license:** I recommend releasing your open source research software with an OSI-approved license to ensure that it can be "freely used, modified, and shared" (https://opensource.org/licenses). Otherwise, you won't gain any of the benefits discussed earlier in this editorial.
- **Carefully choose your repository host:** Journal articles are archived and may be read far into the future. If a future reader follows a link to your source code, you want them to find it. Any of the major services, such as GitHub, GitLab, or BitBucket, are likely to be around for the foreseeable future. Additionally, they have active communities, which is beneficial for gaining new collaborators.
- **Use an archiving service:** Actively maintained source code repositories are fluid. Consider using a service such as Zenodo or Figshare to archive a snapshot of your repository at the time of your journal article. This guarantees that the exact version of your code as of the publication of your journal article remains available to readers. Zenodo is integrated with GitHub, making archiving simple. Another option, essentially with the same effect, is to generate a GitHub release at the time of your journal article.
- **Link to your article:** In your source code repository, show your article citation, and link to it via the DOI, so users of your software know how to cite it.

References

Haider, Z., Hussain, R., Khan, I.L., Shakeel, A., Ijaz, B. and Malik, S.A. (2017) Evaluation of capabilities of open source cognitive radio network simulators. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*: 1814–1817. doi:10.1109/IWCMC.2017.7986559.

Hong, N.C., Hole, B. and Moore, S. (2013) Software papers: Improving the reusability and sustainability of scientific software. doi:10.6084/m9.figshare.795303.v1.

Katz, D.S., Niemeyer, K.E. and Smith, A.M. (2018) Publish your software: Introducing the journal of open source software (joss). *Computing in Science & Engineering* **20**(3): 84–88. doi:10.1109/MCSE.2018.03221930.

Smith, A.M., Niemeyer, K.E., Katz, D.S., Barba, L.A., Githinji, G., Gymrek, M., Huff, K.D. *et al.* (2018) Journal of open source software (joss): Design and first-year review. *PeerJ Computer Science* **4**: e147. doi:10.7717/peerj-cs.147.

Warner, J. (2018) *Thank You for 100 Million Repositories*. Blog, GitHub. URL https://github.blog/2018-11-08-100m-repos/.