Review of "The Four Pillars of Research Software Engineering"

Silvio Peroni[1]

1 University of Bologna

Metadata

Title: The Four Pillars of Research Software Engineering

Author: Jeremy Cohen, Daniel Katz, Michelle Barker, Neil Chue Hong, Robert Haines, Caroline Jay

Submitted to: IEEE Software

Review

In this article, the authors propose that research software engineers (RSEs), a role which is well-known in research teams in several STEM (science, technology, engineering, and mathematics) and SSH (social sciences and humanities) disciplines, should be recognised appropriately by and within the academia, while nowadays there are no particular incentives that permit such RSEs to have a proper academic career path. To address this issue, the authors introduce a theoretical model based on four fundamental “pillars” (software development, community, training, and policy) that would enable the formal recognition of RSEs and their work within an academic framework. Real examples of people, institutions, and activities that have worked along the direction sketched by the pillar accompany the discussion.

The article is well written and shares an essential vision about the role of research software engineers and the importance of research software in any activity related to research. Thus, I firmly believe that such “manifesto” fits very well in IEEE Software. However, the authors should address some issues before publication and, consequently, another round of review could be necessary.

As a side note: all the following comments are thought by considering an Open Science view on the RSE topic. This view is reasonable in this context (i.e. in academia), where the advances in science should be common goods and, as such, should be openly available to the public. Of course, this personal view may crash in some scenarios, in particular when applied science to industrial interests are in place, but it should generally be a valid norm in academia.

Software development encounters the (research) community via early software releases

Indeed, the tools available to software engineers – development methodologies, documentation, code management – are crucial for building maintainable, sustainable, correct, and robust software. However, there is another habit that is important when developing software and that should be introduced and even discussed in the article, considering that in principle could be felt as counter-intuitive: “Release early. Release often. And listen to your customers.” (from ‘The Cathedral and the Bazaar’ by Eric S. Raymond, http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html).

Indeed, the fact that one releases research software (and, crucially, its sources) in its early stages – i.e. when it is still incomplete, incorrect in some aspects, not well-documented yet – could be a substantial added value. In particular, early releases could enable one to understand the needs that may derive from research projects which are similar to the one for which the RSE is working on for his/her commitments. Besides, such releases can even foster possible and unpredictable collaborations with RSEs of other research groups, which may result in quicker debugging, cooperative collection of requirements, and parallel development.

This aspect is not only a technical one, but it also involves the research community as a whole. Thus, it should stand somehow on both the first two pillars which concern software development and community.

Towards recognition of research software at the institutional level

To properly recognise RSEs, software should become an artefact which is formally considered a countable and assessable product of research like the research articles – the primary and first kind of research outcome considered in one’s academic career – and the data used for drawing the specific conclusions in such articles. Of course, the social pressure done by groups such as the Software Sustainability Institute (UK) and the US Research Software Sustainability Institute (US) is crucial to convince the institutions which handle the career of RSEs (e.g. government and universities) to start to consider such new product of research.

However, such social pressure is not enough for reaching this goal, but we also need a technical and infrastructural commitment, e.g. for collecting and preserving this new kind of research product, for enabling its reuse, and for assessing its intrinsic quality and its overall usefulness. Luckily, in the past years several initiatives and events have started to work in this direction, such as Software Heritage (https://www.softwareheritage.org/), FORCE11's Software Citation Principles (https://www.force11.org/software-citation-principles), the Zenodo + GitHub alliance to make a code citable (https://guides.github.com/activities/citable-code/), the Essential Open Source Software for Science call by the Chan-Zuckerberg Initiative (https://chanzuckerberg.com/rfa/essential-open-source-software-for-science/), and the eLife Innovation Sprint meetings (https://sprint.elifesciences.org/). Thus, the authors should integrate the discussion on Pillar 4: Policy with these and other similar aspects, since they are crucial for getting to the official recognition of software as a formal part of the academic research framework.

Pillars 1-2-3 vs Pillar 4

The authors recognise that the implementation of Pillar 4 "can provide significant top-down impact to support the other 3 pillars". While agreeing with this claim, defining the policy framework a pillar contrasts with the importance the authors are providing to this aspect. If the goal is to recognise formally the work done by RSEs, to enable them a proper career in academia, thus that policy should be a kind of ground for all the other three pillars.

However, the scenario could be, in principle, more complicated. On the one hand, no policy means no recognition and, if there is no appropriate recognition of the RSE's work, he/she will not be willing in investing time and effort in amplifying his/her knowledge and skills for implementing the other three pillars. On the other hand, the process of convincing policymakers that software should be a first-class research entity must pass through the added value provided by a complete implementation of the first three pillars. This situation may appear like a chicken or the egg dilemma, but this dichotomy should be discussed by the authors, who should also provide some workarounds or even solutions to make the whole situation acceptable, working, and eventually implementable in the medium/long term.

References missing

This article is as a sort of manifesto to help RSEs and, in particular, policymakers to work together to reach the common goal of creating a framework based on the four pillars introduced by the authors. Thus, in order to be useful for this aim, it would be essential to add as many references as possible to key documents that clarify the aspects mentioned in the article that can be obscure to a reader, who may not have the technical knowledge to understand all the terms mentioned in the article. Just a few examples:

- Software citation: Smith, A. M., Katz, D. S., Niemeyer, K. E., & FORCE11 Software Citation Working Group. (2016). Software citation principles. PeerJ Computer Science, 2, e86. https://doi.org/10.7717/peerj-cs.86
- Software preservation: Di Cosmo, R., & Zacchiroli, S. (2017). Software Heritage: Why and How to Preserve Software Source Code. IPRES 2017 - 14th International Conference on Digital Preservation, 1–10. Retrieved from https://hdl.handle.net/11353/10.931064
- Open software: Fogel, K. (2015). Producing Open Source Software—How to Run a Successful Free Software Project. Retrieved from https://producingoss.com

Other points

- The text in the bottom part of Figure 1 is too small, and it is a bit difficult to read.
- Avoid using an in-text reference pointer (e.g. “[7]”) as the subject of a sentence (“[7] looks at…”). In these cases, please rephrase as “In [7], Nangia and Katz look at…”.
- The examples provided in section 4 as support of each pillar are a bit English-centric, probably due to the authors’ experience in the topic derived from their affiliations (UK, US, Australia). Honestly, it would be great to see at least a justification for this choice. In particular, are the real examples presented the only ones available, or do there exist other initiatives and events that have been run by other areas, e.g. Europe? In case, one can even think to question pan-European associations (e.g. Informatics Europe, https://www.informatics-europe.org/) which may have a better view of whether similar initiatives are also happening in Europe.