Making open source research software visible: new models for sustaining research software development

Neil Chue Hong
Software Sustainability Institute
15 June 2019
N.ChueHong@software.ac.uk

In “traditional” open source software projects, development is often sustained by creating a community of contributors from different organisations that collectively provide effort towards the ongoing maintenance and feature development of the software. Although there are examples of the same model being used for open source research software, it appears to have a smaller chance of success. In this position paper, I hypothesise that this is due to the differing competing drivers for contributing to software present, particularly in academic settings, and also suggest some models which might improve the contributions.

The argument can be made that open source software is like a forest where anyone is allowed to cut trees for timber or firewood. If no-one helps plant new trees, eventually the forest will disappear. Therefore there is an argument that can be made to organisations using open source that it is in their interests to contribute back so they can continue using that software.

The challenge is that there are additional drivers for research software in academia. The curse of novelty is one which is well identified, where it is easier to get funding for producing new things than for maintaining or reusing existing work. Likewise, there is a challenge because the main driver for the use of the software is not the software itself, but the research output it enables. However both of these can be seen elsewhere, and I do not believe these are the main challenges.

Instead, I argue that the biggest challenge is that it is harder to make the “forest” argument, because there is a larger disconnect between the people benefitting from the software and the people controlling the budgets. Paying money for software is commonplace in universities: for things like email, research services like Web of Science, and large academic software packages like Matlab, Qualtrics and SPSS. What is not commonplace is paying for open source software, because the use of such software is split across many more people, for many different reasons and it make it much harder for the true impact of the software to be understood.

I propose a number of recommendations to make this impact more visible, and put it at the heart of what universities do:

1. As universities setup research software engineering groups, broker an agreement with RSE group leaders to set aside a set percentage of their time to contribute to open source software that is important to the organisation. For this to work, the way they use that time must be prioritised by the projects they are working on, based on what they are best placed to contribute to, not by their university (who only get to choose which software projects they are contributing to).
2. Universities should seek to associate their brand with open source software more. Universities contribute significant amounts of effort already to leading open source projects, and yet it is often unrecognised.
3. Universities should promote working on open source projects as part of Undergraduate and Masters programmes, similar to doing internships, so it is seen as valuable career development for both students and faculty.
4. University libraries should consider subscription style donations to some of the most significant open source research software projects that they rely on, as determined by annual surveys.

The challenge with maintaining research software is that there is more software produced than we need to maintain, but there is more software that needs to be maintained than we are currently doing. By making software more visible, it becomes easier to make the argument that we can't keep taking from open source software projects without giving back.