Software Sustainability - lessons learned from different disciplines
Date: August 8, 2018
Presented by: Neil Chue Hong (Software Sustainability Institute)

Q. Concerning the reluctance of developers or PIs in sharing their software, it could be related to putting professional reputation at risk or lack of funding. Sharing as a funding criterion might help. Could you comment on that?

A. I personally think this is a good idea. We have seen an increase in the sharing of publications and data since directives around open access and open data / data management plans have been implemented in the UK, and I believe similar approaches would help for software. However as well as the "stick", there needs to be the "carrot" and we need to provide incentives for sharing as part of funding frameworks, perhaps by increasing the scoring of proposals (in the engagement and impact categories) who have solid research output management plans that include the publishing of software and code (as well as other research outputs).

Q. What about hiring software engineers and training them in the necessary science?

A. I think this is a valid approach, particularly in larger organisations and teams. I think there's a broad spectrum of people who work on the "coding" side of research. At one end, there's researchers who "dabble" or "hack" code and at the other end there's software engineers contracted in to provide specific skill sets in a larger research software project. In between, there's an opportunity for software engineers to be trained not just in the necessary science, but more importantly in the scientific process and research methods. Anecdotally, the trickiest thing is getting software engineers used to the different workflow in research, although this will depend on their background - it is not dissimilar to particular software development models.

Q. Follow on to first question. Do you believe that such codes should be copy-left open source licensed to enforce sharing? Would that be beneficial or harmful?

A. I personally believe that codes should be permissive rather than copy-left as I think the lowering of barriers to reuse is more important that enforcing sharing. The other issue is that typically the enforcement of sharing is quite hard if someone contravenes it and you don't have a good legal team behind you. So whilst this might work for PIs and research project leaders at larger institutions, for those at smaller ones they may have no effective way of enforcing the terms of a copy-left license. This article by Jake VanderPlas gives a good summary of the pros and cons of each approach for different sizes of projects: http://www.astrobetter.com/blog/2014/03/10/the-whys-and-hows-of-licensing-scientific-code/

Q. I agree that educating our peers in sustainable practices is a vital need. Do you have any advice for groups comprised of senior members who have domain expertise and junior members who have computer science training? Bi-directional knowledge transfer across generational divides and through disciplinary ontologies is quite challenging. Have the survey's captured these cultural and disciplinary dynamics?

A. The surveys that the SSI and our collaborators are just starting to understand the cultural and disciplinary dynamics. In terms of disciplines, one thing that is clear is that each discipline has people and groups who are heavily invested in software, and in computational and data approaches. The differences are around how these people and groups are regarded by the rest of the discipline: are they embraced, seen as heretical, or tolerated with either scepticism or fear that they will become the norm and others will have to retrain.
Generational divides are more interesting. Certainly an issue that has been picked up from our surveys is that supervisors are ambivalent about trying out new software tools or processes. A common question is “if the existing approach worked for me, why do we need to change it?” however this is starting to get pushed back because of the prominence given to the “reproducibility crisis” in the popular press. What we see, delving a little deeper, is that at all levels there are those willing to embrace new approaches to research to help get ahead and there are those who once they get to a certain level are happy to continue without radically changing again. If the evidence from our Fellowship programme is anything to go by, the champions for better software come from across many generations, ranging from those just starting to embark on a PhD career to decades in post Professors.
So what advice can I give to help bridge divides? I think the main thing is to focus on the benefits that using good software techniques will bring to the research that a group is doing, from making it easier to change papers on the day of submission and increasing the number of collaborators and citations your work gets, to making it easier to on-board new members of your research group and tackling larger and more complex research questions.