“Not everyone can use Git”: Research Software Engineers’ recommendations for scientist-centred software support (and what researchers really think of them)

“Not everyone can use Git”: Research Software Engineers’ recommendations for scientist-centred software support (and what researchers really think of them)

Caroline Jay, School of Computer Science, University of Manchester, UK, caroline.jay@manchester.ac.uk; Rawan Sanyour, Department of Information Systems, Taibah University, KSA, rsanyour@taibahu.edu.sa; Robert Haines, School of Computer Science, University of Manchester, UK, robert.haines@manchester.ac.uk.

Abstract

Research Software Engineers (RSEs) are at the coalface of ensuring that computational science is accurate, reliable and reproducible, and their views on making progress in this domain are therefore particularly valuable. This is not always recognized by the scientific community, however: work examining the challenge of developing research software tends to focus on the views of academics, and RSEs’ voices are rarely found in the evidence base.

We report the results of a project that brings together researchers and RSEs, to determine how to help scientists publish their code. We interview scientists to identify challenges, and we interview RSEs to determine how to overcome them. We formalize the results into a set of recommendations, and evaluate them in a survey completed by 65 computational scientists.

The survey shows that improving tool GUIs, linking internal repositories to external ones, and training in version control would all aid scientists in publishing reliable code. It also demonstrates that RSEs' views are valued by researchers: every recommendation received strong support. We conclude that RSEs can play a crucial role in scientific software policy, and their expertise should be officially recognized.

Introduction

Software is now central to science. It is most commonly used for data analysis, but can play a role across the whole research lifecycle, from data-mining for hypothesis generation, to forming an output, such as a scientific model. In spite of its value, however, a great deal of research software remains unpublished (Peng, 2011). This is potentially a huge loss to science: whilst very few papers would exist without the aid of software, software exists in its own right, and may have uses that extend far beyond a single publication (De Souza, 2014). To paraphrase, whilst a paper could not exist without the software, the software can exist without the paper.

Common reasons proposed for a failure to publish code include the fact that it is difficult to run on a generic platform, that it is too time-consuming to document, and that it is not considered of sufficient quality to be made publically available (LeVeque, 2013).

This paper reports empirical work examining the reasons why scientists do not publish their code, and evaluating recommendations from research software engineers (RSEs) for promoting

this activity. We conducted interviews with scientists, which aimed to identify the barriers facing scientists trying to publish their code, then interviewed RSEs, to elicit solutions to these barriers. The proposed solutions were summarised in a questionnaire that received responses from 65 computational scientists. The results show that many researchers find using repositories to be particularly difficult, and would welcome both technical solutions to supporting reproducibility and training in a wide area of software engineering-related practice.

Here we present a summary of the methods and results. Full details of the study, materials, and data are provided in our our GitHub repository: https://github.com/IAM-lab/rse_support_survey

Interviews

As a first step to understanding the barriers to code publishing, we interviewed seven researchers (all male) who were working in a range of computing-related disciplines and actively trying to publish their code. Following this, we interviewed seven research software engineers (four male, three female) who had experience developing and/or helping researchers write scientific software, to obtain their views on how to address these barriers. All participants came from the University of Manchester, and were purposively sampled. The interviews were audio recorded and transcribed, and thematically analysed in an open coding fashion following established analysis methods (Braun, 2006), using NVivo[1] software.

Researchers' experience trying to publish code

The researchers were asked a series of questions in a semi-structured interview, which encouraged them to articulate their views about code publishing, and discuss any difficulties they had with the process.

The researchers shared a conviction that making scientific work reproducible is important. Releasing code to the community was seen to enhance the credibility of work, by demonstrating its accuracy, with benefits for reputation and career. P7 preferred to publish pseudocode (although would be prepared to share code on request), but the other participants endeavoured to make their all their code and data publically available.

'I did everything I could in order to allow scientists at least to reproduce it with the same data, same code and same method' (P4).

'I always thought that my tool had to be open, had to be available… trying to be open and say look I did this, and even if it is a small thing it is true, and it is working' (P1).

Three participants expressed a reluctance to publish code due to it not being high enough quality, and being judged as a result.

[1] http://www.qsrinternational.com/nvivo-product

‘Sometimes I feel embarrassed about my code and sometimes that’s why I don’t disseminate or publish a lot because of that’ (P3).

P2 recognised a positive in this: although exposing yourself to criticism is ‘scary’, publishing code is a good way to verify work, and allows errors and bugs to be fixed before proceeding further. P1 also stressed the benefits of cleaning up code prior to publication in terms of improving software engineering skills, and said that after a while writing good code would become ‘automatic’. P5 argued that if the code is designed to be reused, it should be refined. He also recognised that revisiting code is a good learning experience. P6 says that to avoid extra work, the code should be well documented, up-to-date and well-engineered from the earliest stage. This is a necessary practice for him, as he works with scientists from other domains, who must verify what he has done. Six of the participants recognised that revising code so that other people can use it or understand it is not always a priority, however, and agreed it could be viewed as an overhead. Nevertheless, the dissatisfaction they experienced at not always achieving the level of reproducibility they desired was a strong motivation to improve the way they published code.

Giving competitors an advantage through using their code was not a key concern of participants, and five of them said this would not prevent them from publishing. P2 said that if someone uses his code or data, even in a way that is better than the original work, this is positive because it is beneficial for science. P6 thought that having open code was more likely to encourage collaboration than competition, an idea shared by others:

‘The whole idea of this collaborative approach is that people will improve something incrementally and everyone can have ownership of this’ (P2).

‘If you want to maximize the number of publications in the short term… instead of preparing code for publishing, you will publish a new paper. But if you want to have an impact on the community, the best way is to… make others’ jobs easier by using your methodology. That is the difference between a long term and a short term philosophy’ (P4).

In addition to the overhead associated with ensuring code is suitable for publication, participants identified other barriers that were also primarily technical. P5 discussed the difficulty caused by ‘islands’ of analysis code, analysis plots, datasets and database for the results, which must be ‘stitched together’ in a particular way to reproduce results. Five participants expressed difficulty with using repositories, such as GitHub, and two had difficulty working out how and where to store significant amounts of data.

Research software engineers’ views on supporting code publishing

A second set of questions was prepared, based on the barriers raised by scientists, to serve as the interview schedule for RSEs. In particular, RSEs were asked to think about how these problems - and others they had come across professionally - could be resolved.

The RSEs recognised all of the concerns that the researchers raised. There was a consensus that in many cases the solution was training; for example, there are repositories available for code and data, but researchers often do not know they exist, or do not understand how to use them. To address difficulties reproducing workflows, use of virtual machines or containers was proposed, and to help support repository use, new user interfaces were suggested.

Survey

The suggestions from RSEs were synthesized and phrased as a set of questions within a survey. Sixty-five computational scientists completed the survey, including 21 academics, 18 PhD students, 8 research associates, and 5 software engineers. Respondents were from subfields of computer science, biology and bioinformatics, physics, materials science, medical science, neuroscience, climate change and oceanography. The majority appeared to be in a technical discipline, although it was not possible to infer their precise level of software engineering knowledge. The survey questions, and the results, are reported below. Table 1 covers technical solutions to barriers, Table 2 covers training, and Table 3 contains questions relating to community and support.

Technical solutions	Yes	No	Other
If repositories had a simple, easy to use GUI, alongside the current command line/GUI interfaces, would this encourage scientists to use them?	65%	15%	21%
Would it be helpful for the repository to ‘track changes’ to a script, so they don’t have to be documented manually?	67%	17%	17%
Would you like your institution to have a central repository, which could be used to store all data and code internally?	55%	31%	15%
Would you like a central internal repository to be compatible with an external one, such as GitHub, so any code and data can be published externally when required, and anything updated in an external GitHub repository would automatically appear on an internal one?	88%	8%	3%
Would you be interested in using VM or container technology for your own research?	73%	21%	7%
Would you be interested in using a VM or container technology to rerun other people’s research, or use their analysis code?	83%	11%	6%
Do you think using VM or container technology will have any effect on scientific reproducibility?	72%	15%	14%

Table 1: Survey questions regarding technical support

Which of the following types of training or support would you find it useful for learning to use repositories?	
Practical workshops	60%
Online training	60%
Guidance/details about the various types of repository and their features	52%
A glossary of the terminologies relating to repositories	45%
Other	22%
In which of the following areas would you find training/support helpful?	
Data management planning	67%
Computational research skills	62%
General software engineering skills	56%
Awareness about intellectual property issues	44%
Awareness about open source licensing systems	56%
Other	10%

Table 2: Survey questions regarding training

Community and support	Yes	No	Other
Would having access to research software engineers (developers who specialise in scientific software) on campus be something that could help scientists produce code that they would be happy to publish?	76%	5%	19%
Would you want to become part of a research software engineering community, where people could discuss code publishing and open science?	65%	20%	15%
Would you be more inclined to focus on making analysis code and data available if more publishers required it?	77%	18%	5%

Table 3: Survey questions regarding research software engineering support

Discussion

Software is changing the way we conduct science, in terms of the sophistication of the analyses we perform, and the volume of data we can process. It supports real documentation of the research process, making reproducibility genuinely possible, and improving the accuracy of the results. By making methods, models and analyses open to others, we can massively accelerate the rate at which we gather knowledge and make discoveries.

The interviews reported here indicate that software is also having a cultural impact: it is changing the way we look at science, leading to an openness to sharing and an understanding that this is the best way to make scientific progress.

Overall, the study provides recommendations that can be used to help support scientists trying to publish their code. It should be noted that the sample was small and may not be representative of the wider research community. Nevertheless, the results still contribute to the evidence base for understanding how to tackle the issue of 'hidden code'.

As well as providing data that can potentially be used to improve research software, the work also demonstrates that scientists overwhelmingly support the solutions proposed by RSEs. The RSE community is only just becoming established, and its role is still evolving, but the knowledge and expertise of RSEs mean they are ideally placed to help solve challenging problems, such as the failure of scientists to publish code. This study provides evidence of the direct value they provide to this area, and indicates that involving RSEs in decisions about research software from both a practical and policy perspective could make a significant contribution to the advancement of science.

References

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, *3*(2), 77-101.

De Souza, M., Haines, R. & Jay, C. (2014). Defining Sustainability through Developers' Eyes: Recommendations from an Interview Study. 2nd Workshop on Sustainable Software for Science: Practice and Experience.

LeVeque, R. J. (2013). Top Ten Reasons to Not Share Your Code (and why you should anyway). *SIAM News, April*, *1*.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, *334*(6060), 1226-1227.

This is a pre-print, pre-final version of a paper invited to appear in the RSE UK special issue of JORS, 2017.