Software in Reproducible Research: Advice and Best Practice collected from experiences at the Collaborations Workshop

Shoaib Sufi[1], Neil Chue Hong[2], Simon Hettrick[3], Mario Antonioletti[2], Stephen Crouch[3], Alexander Hay[3], Devasena Inupakutika[3], Mike Jackson[2], Aleksandra Pawlik[1], Giacomo Peru[2], John Robinson[3], Les Carr[3], David De Roure[4], Carole Goble[1], and Mark Parsons[2]

School of Computer Science
University of Manchester
Oxford Road
Manchester, M13 9PL
United Kingdom

EPCC, School of Physics
University of Edinburgh
JCMB, Mayfield Road,
Edinburgh, EH9 3JZ
United Kingdom

Web and Internet Science
University of Southampton
Highfield Campus
Southampton, SO17 1BJ
United Kingdom

Oxford e-Research Centre
University of Oxford
7 Keble Road
Oxford, OX1 3QG
United Kingdom

Corresponding Author Email Address: S.Sufi@software.ac.uk

ABSTRACT
The Collaborations Workshop 2014 (CW14) brought together representatives from across the research community to discuss the issues around software's role in reproducible research. In this paper we summarise the themes, practices and ideas raised at the workshop. We also consider how the "unconference" format of the CW14 helps in eliciting information and forming future collaborations around aspects of reproducible research. In particular, we describe three distinct areas of concern which emerged from the event: collaboration readiness, capability enhancement and advocacy.

Categories and Subject Descriptors
D.2.4 [**Software Engineering**]: Software/Program Verification – *reliability, validation.* D.2.5 [**Software Engineering**]: Testing and Debugging – *code inspections and walkthroughs.* D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancements – *documentation, restructuring, reverse engineering, and reengineering, version control.* D.2.8 [**Software Engineering**]: Metrics – *product metrics, software science.* D.2.9 [**Software Engineering**]: Management – *copyrights, life cycle, programming teams.* D.2.13 [**Software Engineering**]: Reusable Software – *reusable libraries, reuse models.*

General Terms
Management, Documentation, Design, Reliability, Human Factors, Legal Aspects.

Keywords
Software sustainability, reproducible research, reproducibility, Software Sustainability Institute, Collaborations Workshop.

TRUST'14, June 09-11 2014, Edinburgh, United Kingdom
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2951-4/14/06…$15.00.
http://dx.doi.org/10.1145/2618137.2618140

1. INTRODUCTION
A significant (and increasing) amount of research is fundamentally reliant on results generated by software. Regardless of this fact, software is infrequently subjected to the same scientific rigour as is applied to more conventional experimental apparatus. Consequently, the best practices required for reproducibility are often overlooked with software-based research, making it difficult if not impossible to reproduce and verify results generated by software.

On 26-28 March 2014, the Collaborations Workshop (CW14), run by the Software Sustainability Institute (SSI) [1] brought together a group of representatives from across academia and industry: researchers, software developers, project managers and funders. The workshop was themed around the topic of *"the role of software in reproducible research"* and allowed the experiences and ideas of all attendees to be discussed and recorded so that the combined expertise of the workshop attendees could be summarised into a set of output: themes, best practice for software reproducibility and ideas to help improve reproducibility.

For the event, we purposely did not provide a strict definition reproducibility as this is still a developing term[1].

2. OVERVIEW OF THE WORKSHOP
Ensuring that research is a complex, multi-faceted and sometimes subjective problem. The CW14 used a format based around the "unconference" model (itself based on Open Space Technology techniques [2]) that enabled the 60+ attendees to all have a voice in the discussion.

The CW14 employed a range of techniques to elicit information from attendees and disseminate it across the workshop as a whole. Lightning talks provide an opportunity to discuss an issue of importance to the presenter, but are limited to two minutes, making them an efficient use of the workshop's limited time. Such a short time also forces the presenter to focus only on the main issue that concerns them. It also ensures that all participants

[1] David De Roure's talk at the 37th UKSG Annual Conference

have a basic understanding of each others' work, experience and position in relation to the topics.

Rather than setting an agenda in advance, with the organisers ultimately dictating what will be discussed, in an unconference topics are solicited from the attendees and consensus reached during the event on what will be discussed next. This is not only a flexible way of extracting the main issues from a large group of people, it also ensures that the agenda represents the views and interests of the attendees, which encourages their participation.

Once the agenda has been set, the attendees split into groups of up to ten people to discuss the topics on offer. Each group is given an hour to discuss the topic and draw out a summary and recommendations. Once the hour is up, all groups meet and take it in turns to present their findings back to all attendees.

Similarly, "collaborative ideas" sessions bring attendees together in small groups with the focus on using their amassed skills to identify and propose solutions to an issue related to reproducibility. Finally, an associated "hackday" at the end of the event was an opportunity for developers and researchers to implement prototypes based on some of the ideas that came from the rest of the event.

It is of vital importance that the discussions and conclusions from the workshop are made available to the research community at large. Consequently, everything discussed at the workshop is quickly added to the workshop website to ensure a permanent and easily accessible record is maintained[2].

3. THEMES ARISING FROM CW14

Overall, there were 14 discussion topics considered and 22 collaborative ideas generated at the CW14. Some topics were strategic, some tactical and many practical. This reflected those who were attending: many of them practitioners of research or development who were keen to learn from others and from the thought leaders in the area who provided keynote presentations. They also had the ability to become voices within their own communities to promote best practice in reproducible research.

From the many discussions and recommendations, three definite themes emerged:

- Collaboration Readiness
- Capability Enhancement
- Advocacy

For each theme we examine the best practice collected and the ideas for the future developed.

3.1 Collaboration Readiness

3.1.1 Best practice

A "shopping list" was created of tools and technologies based on attendee suggestions that might have benefits to the development of research software. These included GitHub[3] for collaborating on software development, Figshare[4] for basic sharing of data, *make* for command-line automation of tasks and updating analysis painlessly if underlying data had been updated, and notebook and workflow systems such as IPython Notebook [4] and Galaxy [5] to help re-play and reproduce analysis work.

It was also noted that the software used in research should adhere to certain quality constraints that increase the confidence in the software and increases its use. These include that it: should work; should be documented well enough to describe the function of the software, and enable others to produce comparable results; should have a test suite; and that the source code should be available. This echoes other work in this area such as the Five Stars of Research Software [3], and Code as a Research Object[5].

3.1.2 Ideas for the future

There was a large focus on quality of software and how this was a proxy for reproducibility. This included suggestions of automated and manual crowd-sourced annotation of software source code to arrive at an understanding of how "good" the code was[6]. This in turn gives an indication to others of how understandable, and therefore trustable, a piece of code is.

Open Source Health Check[7], the winner of the CW14 hackday competition, assessed whether a software repository had readme files, license files and continuous integration configuration files and would alert the maintainers (by automatically creating an issue) where these important pieces of documentation were missing.

There were also ideas to connect the disparate sources of information involved in research together - e.g. code, lab books, software, data and parameters, figures and papers and proposals. Work on Research Objects[8] was one suggestion that might further support reproducibility via these sources.

3.2 Capability Enhancement

3.2.1 Best practice

The attendees noted that electronic notebook systems such as IPython Notebook and R Studio[9] should be used as they bring an openness to the process, tools, versions and data one is using; however they are not a panacea and problems can still arise in analysis (i.e. errors).

The use of virtual machines (VMs) to package reproducible computational experiments was highlighted and best practice of configuration such as the link between cores and virtual disks was discussed. Getting the research community to do this, as well as creating simpler interfaces for them, was seen as a key way of engineering VM based systems that try to cater for reproducible

[2] Agenda: http://www.software.ac.uk/collaborations-workshop-2014-cw14-software-your-reproducible-research/cw14-agenda

[3] Github: http://www.github.com/

[4] Figshare: http://www.figshare.com/

[5] Code as a Research Object: https://github.com/mozillascience/code-research-object

[6] Hacker or Slacker: https://github.com/hogliux/hackerorslacker

[7] Open Source Health Check: https://github.com/OpenSourceHealthCheck/healthcheck

[8] Research Objects: http://www.researchobject.org/

[9] RStudio: http://www.rstudio.org/

research infrastructure and catalogues of reproducible research, such as Recomputation.org[10].

On the human side it was clear that developers who specialise in helping researchers will always be needed as even with some software development skills, the researchers focus is research and the developers focus is building tools to aid the research.

Programming languages that support easy understandability, better documentation, modularity, integration of test code, versioning and access to codes written in other languages / systems were favoured in terms of better supporting reproducibility: some examples mentioned were F#'s ability to encode the intent of types e.g. dimension types, and the Go-lang's in-built package management and enforced code style guide at compile time. Community infrastructure (forums, libraries, availability and support) were seen as key to the 'features' of a language and its support for reproducibility; the implication being that it itself would be sustained and thus not a fragile dependency.

3.2.2 Ideas for the future

Ideas included making tools easier to use for non-experts such as visual versions of *make* which made it easier to re-run your analysis if the dependencies changed, and to hide version control systems and repositories behind the GUI tools used by researchers (e.g. a genome annotation application where the "master" copy of the data was available as a GitHub repository was suggested as a collaborative idea).

The digitisation of chemical structure was given as example of how written lab book based work could be made more digital and therefore available, linkable, and usable in reproducible research. The ability to turn spreadsheets into web service accessible data sources was also suggested: this was seen as making them more useful by allowing programmatic access.

3.3 Advocacy of Reproducible Research

3.3.1 Best practice

Training was seen as key. This included training in computational competencies by way of Software Carpentry [6] for PhD and early career researchers but also to educate current project leaders and Principal Investigators into the advantages of reproducible research. In addition Open Science training [7] and the advent of Massive Open Online Courses (MOOCs)[11] were seen as mechanisms to help get researchers trained and informed. An ongoing challenge is how to get training ingrained in research culture, and show a more direct benefit towards reproducible research. One suggestion for possible best practice was to get learners to reproduce other people's work as part of the training.

The issue of funding dedicated software developers was also a focus. From the nuts and bolts of various funding models that one can use on Research Councils UK and European Commission funded projects to the justification needed and use of "pathways to impact" for software, experience was shared. Providing examples of successful grants for others to inspect, and naming appropriate reviewers who are respected and independent but are known to understand the need for developers being funded were suggested as ways of improving practice in this area.

[10] Recomputation.org: http://www.recomputation.org/

[11] Wikipedia definition of MOOC: http://en.wikipedia.org/wiki/Massive_open_online_course

The SSI have supported the creation of a title: *"Research Software Engineers"* for those who are working to support research. This term has been favoured due to its increasing ubiquity, established meaning and nascent community[12].

3.3.2 Ideas for the future

One of the telling phrases at the CW was *"Reproduciliteracy"* which was proposed having material available to promote the benefits of reproducible research to the communities of those represented at the workshop. Supporting this was the idea of a "Hacker News"[13] for the Open Science community. As much is written about reproducible research in blogs and on various sites, some type of aggregator or digest highlighting the most important recent discussions would enable interested people to keep up to date efficiently.

There was also a focus on developing standards for citing software. Traditionally this tends to be done by citing a paper about the software but this approach was novel in that it cited the software directly and produced a format which worked with existing systems such as EndNote. Another idea was building systems that help track how long software takes to develop such that this can be used as evidence in funding proposals and to give more credit for those who write software to support research.

One of the impressive ideas developed during the hackday was updates to ScienceToolBox[14], a tool which automatically trawls through Google Scholar and GitHub to link developers with software and that software with its citations: normally a manual process. In the future, linking tools like this to products like ImpactStory[15] will help to show the contribution that developers who write software make to research outcomes.

4. CONCLUSIONS

It is clear that collaboration readiness, capability enhancement and advocacy are key themes in reproducible research. What has become apparent is the large amount of work still to be done to socialise these ideas in the various research domains. Only by doing this can the collective community bring about the much needed changes in norms so that the contributions of software and data is properly recognised. Likewise, the best practice identified in the CW14 will only become commonplace if we can show the benefits.

The perennial problem of motivating researchers and those who support them to put in the effort to make their work reproducible is key: this requires a culture change, and culture changes take time. However investing in education, systems of credit for software, advocacy, training of researchers in computational techniques and mandating of publishers to require code and data deposition can be seen as ways forward. While we have many good ideas for collaboration readiness and capability enhancement, continued advocacy is needed to help bring about the improvements which the community of software in research know are needed to make research more reproducible.

[12] Research Software Engineer community: http://www.rse.ac.uk/

[13] Hacker News: https://news.ycombinator.com/

[14] Science Toolbox: http://sciencetoolbox.org/

[15] Impact Story: https://impactstory.org/

5. ACKNOWLEDGMENTS

The authors and the Collaborations Workshop 2014 event were supported by the UK Engineering and Physical Sciences Research Council (EPSRC) Grant EP/H043160/1 for the UK Software Sustainability Institute. We also acknowledge the support of Microsoft Research, GitHub and the Oxford e-Research Centre who provided sponsorship of the Collaborations Workshop 2014.

6. REFERENCES

[1] Crouch, S., et al. 2013. *The Software Sustainability Institute: Changing Research Software Attitudes and Practices,* Computing in Science & Engineering, vol.15, no.6, pp.74-80. DOI: 10.1109/MCSE.2013.133.

[2] Owen, H. 2008. *Open Space Technology: A User's Guide (3rd ed.)*. Berrett-Koehler. ISBN: 978-1-57675-476-4.

[3] Chue Hong, N. 2013. *Five Stars of Research Software*. Blog post accessed 5th May 2014 from http://www.software.ac.uk/blog/2013-04-09-five-stars-research-software

[4] Perez, F., Granger, B.E. 2007. *IPython: A System for Interactive Scientific Computing*. Computing in Science & Engineering, vol.9, no.3, pp.21-29. DOI: 10.1109/MCSE.2007.53.

[5] Sandve, G.K., Nekrutenko, A., Taylor, J., Hovig, E. 20XX. *Ten Simple Rules for Reproducible Computational Research*. PloS Computational Biology, vol. 9, no. 10, e1003285. DOI: 10.1371/journal.pcbi.1003285.

[6] Wilson, G. 2014. *Software Carpentry: lessons learned*. F1000Research, vol. 3, no.62. DOI: 10.12688/f1000research.3-62.v1.

[7] Kershaw, S. 2013. Open Science Training Initiative: Post-Pilot Report. Technical Report accessed 5th May 2014 from http://opensciencetraining.com/OSTI-PostPilotReport.pdf