Building a Sustainable Structure for Research Software Engineering Activities

Jeremy Cohen*, Daniel S. Katz‖, Michelle Barker‡, Robert Haines§ and Neil Chue Hong¶
*Department of Computing, Imperial College London, London, UK. Email: jeremy.cohen@imperial.ac.uk
‖NCSA, CS, ECE, iSchool, University of Illinois Urbana-Champaign, Urbana, IL, USA. Email: d.katz@ieee.org
‡Australian Research Data Commons, Cairns, Australia. Email: michelle.barker@unimelb.edu.au
§Research IT, University of Manchester, Manchester, UK. Email: robert.haines@manchester.ac.uk
¶Software Sustainability Institute, EPCC, University of Edinburgh, Edinburgh, UK. Email: N.ChueHong@software.ac.uk

***Abstract*—The profile of research software engineering has been greatly enhanced by developments at institutions around the world to form groups and communities that can support effective, sustainable development of research software. We observe, however, that there is still a long way to go to build a clear understanding about what approaches provide the best support for research software developers in different contexts, and how such understanding can be used to suggest more formal structures, models or frameworks that can help to further support the growth of research software engineering. This short paper provides an overview of some preliminary thoughts and proposes an initial high-level framework based on discussions between the authors around the concept of a set of pillars representing key activities and processes that form the core structure of a successful research software engineering offering.**

***Index Terms*—research software, software sustainability, reproducible research, research software engineering**

I. Introduction

While researchers have been building software to support their research for many decades, their primary goal is generally their research outputs, not the software. There has, however, been significant growth in the number of individuals who are interested in the development of the software itself and the process of working with researchers to help design and build quality, sustainable software. This has led to the fairly recent emergence of the concept of Research Software Engineering (RSE). Developed out of discussions at the UK Software Sustainability Institute's [1] Collaborations Workshop in 2012 [2], [3], the concept builds on the fact that developing research software requires increasingly advanced skill-sets that must be built up over time by individuals who specialise in the process of writing code and the application of best practices to ensure its reliability and sustainability. Jiménez et al. [4] provide an example of four such best practices.

As research software engineering has grown, it has become clear that there are several activities that are common between research software offerings at different institutions. It is also clear to us that research software engineering is, or should be, about a lot more than individuals writing research software. This short paper is intended to stimulate discussion around this area and to represent the initial step in defining a multi-pillar framework setting out the activities that we consider are important for a holistic research software engineering offering.

II. Core elements of RSE

Based on our observations of the way research software engineering has developed over the past few years, we see a series of activities that we believe contribute to an institution (or group of institutions) being able to provide an RSE offering to its research community that is sustainable and manageable over the long-term. These activities can be combined into groups covering specific areas which we define as the pillars of research software engineering—the key structures around which a successful research software development capability can be built. These pillars are: **software engineering**, **community**, **training** and **policy**.

It is our assertion that to offer comprehensive research software support, activities from each of these pillars must be provided. We now highlight key elements of the four pillars:

Software Engineering: The process of building software in a research environment is often somewhat different to that which professional software engineers are likely to be familiar with. For example, software in scientific research is generally developed to solve a specific research challenge meaning that it is often built without thought for its longer-term, wider use or maintenance, as highlighted in the work of Morris and Segal [5]. Research software engineers (RSEs) need to interact extensively with researchers to understand what they want to achieve and identify the best way to address their requirements. They generally have a unique and valuable skill set covering research as well as software knowledge which is why they are so vital to this element of the framework.

Community: Communities provide a forum through which RSEs can meet, network, and share ideas and technical knowledge. The value of this cannot be underestimated, especially when working as part of a small development team or research group, or working independently as the only developer within a project or group of non-computational researchers. We also see scope for domain-specific communities to develop.

Training: Training is vital to support skills growth and long-term retention and dissemination of knowledge. There are various mechanisms for providing training which may be general or more domain focused. Training is important at all levels from basic through to advanced-level training for experienced

978-1-5386-9156-4/18/$31.00 ©2018 IEEE
DOI 10.1109/eScience.2018.00015

developers. Ultimately training can help to improve research and software productivity/quality, and software robustness.

Policy: Policy advances are critical to enabling the broader system changes required to increase understanding of software's crucial role as enabling infrastructure for research, and to promote it as a key component of research and cyberinfrastructure strategic planning. We can learn from domains where software is already a key part of the research pipeline. The sort of advances that we hope may be achieved through advocacy and cultural change among institutions, funders, and the wider research community include: Recognising software outputs as first-order research assets and providing means to assign credit to them; Providing better structured career pathways to help sustain RSE roles and improve opportunities for career progression; Incorporating RSEs in funding guidelines; Providing researcher access to RSEs, which can include providing physical spaces for collaboration (see [6]). Measurement is another key element of system change, providing evidence to decision-makers of the benefits of change, and analysing priority areas. Existing survey work on RSEs is already providing valuable confirmation of their key role, and studies on the critical role of software in research add to the argument [7].

III. A sustainable research software framework

Our preliminary structure is a work-in-progress, intended to stimulate discussion and seek feedback with a view to developing a more complete and refined description of the framework. We believe the pillars introduced in Section II:

1) encapsulate the wealth of processes, topics and activities that make up research software engineering, and
2) represent, in their naming, the core, top-level concepts that individuals can identify with as being of utmost importance in enabling research software engineering.

Other aspects required to complete the framework are:

- Defining the processes that link or bring together the concepts represented by the pillars and the wider research environment. This requires an understanding of the synergies between activities that fit within different pillars.
- Identifying the profiles of the individuals that each pillar relates to or targets, e.g. where do researchers, academics, software engineers, RSEs, research data specialists, research managers, etc. fit within the framework? How significant is this in ensuring its success?
- Identifying how generally applicable the current pillar definitions are. Do they apply differently in the context of individuals or groups? How robust are they in the light of possible structural changes in RSE communities?

These points need to be addressed to complete the initial structure. However, this will then need refining. More general questions that we feel will require further investigation/discussion as part of this refinement process are:

- Do these four pillars represent the complete picture? Should further pillars be defined?
- Are any of the topics covered by existing pillars of sufficient importance/significance that they should be promoted to form separate pillars in their own right?
- Is the naming of the pillars correct? Could they be renamed to clarify their meanings or the way they are viewed from different perspectives?
- Are there differences in the way that researchers or RSEs identify with the pillars?

We hope to investigate these issues with the wider research software community to gather feedback and suggestions that help to test, refine, and complete the preliminary design.

IV. Conclusions and Future Work

While research software engineering has come a long way in the past six or so years, it is still in its infancy as a discipline. We have observed that there is still a lack of significant formal structures or models that can be used to explain how and why RSE works in different contexts and, most importantly, how it can be effectively sustained and grown to support a much wider range of researchers. We have provided an overview of a framework that can offer one formalisation of a structure for research software engineering that brings together a full set of activities that we believe are necessary to provide a sustainable offering. This paper seeks to gather feedback and thoughts on the perceived correctness of our proposed structure and suggestions on how it might be improved. Going forward, we intend to prepare a more detailed publication that defines our next iteration of the framework, addressing the various issues raised here. We are keen to open up a wider discussion on this topic and the ideas presented. You can engage with this process by submitting thoughts or questions to the *rse-models* repository [8].

Acknowledgements

JC acknowledges EPSRC grant EP/R025460/1. RH acknowledges University of Manchester support. NCH acknowledges EPSRC, ESRC, and BBSRC via grant EP/N006410/1.

References

[1] The Software Sustainability Institute. https://software.ac.uk. Accessed 2018-06-18.

[2] Software Sustainability Institute Collaborations Workshop 2012. https://www.software.ac.uk/cw12. Accessed 2018-06-20.

[3] R. Baxter, N. Chue Hong, D. Gorissen, J. Hetherington, and I. Todorov, "The Research Software Engineer," in *Digital Research 2012*, Oxford, UK, Sep. 2012, Presentation. [Online]. Available: http://purl.org/net/epubs/work/63787

[4] R. C. Jiménez *et al.*, "Four simple recommendations to encourage best practices in research software," *F1000Research*, vol. 6, no. 876, 2017. [Online]. Available: https://doi.org/10.12688/f1000research.11407.1

[5] C. Morris and J. Segal, "Some challenges facing scientific software developers: The case of molecular biology," in *2009 Fifth IEEE International Conference on e-Science (e-Science)*, Dec 2009, pp. 216–222. [Online]. Available: https://doi.org/10.1109/e-Science.2009.38

[6] The Moore-Sloan Data Science Environments: New York University, UC Berkeley, and the University of Washington. Creating Institutional Change in Data Science. http://msdse.org/creating_institutional_change.html. Accessed 2018-07-05.

[7] U. Nangia and D. S. Katz, "Understanding software in research: Initial results from examining nature and a call for collaboration," in *2017 IEEE 13th International Conference on e-Science (e-Science)*, Oct 2017, pp. 486–487. [Online]. Available: https://doi.org/10.1109/eScience.2017.78

[8] rse-models: Collecting ideas, content, information, feedback and queries on the development of standardised models for sustainable approaches to RSE. https://github.com/jcohen02/rse-models. Accessed 2018-07-09.