# Addressing Research Software Sustainability via Institutes

Daniel S. Katz*, Jeffrey C. Carver†, Neil P. Chue Hong‡, Sandra Gesing§,
Simon Hettrick¶, Tom Honeyman‖, Karthik Ram**, and Nicholas Weber††
*NCSA & CS & ECE & iSchool, University of Illinois, Urbana, IL, USA, ORCID: 0000-0001-5934-7525
†Department of Computer Science, University of Alabama, Tuscaloosa, AL, USA, Email: carver@cs.ua.edu
‡Software Sustainability Institute & EPCC, University of Edinburgh, Edinburgh, UK, ORCID: 0000-0002-8876-7606
§Center for Research Computing, University of Notre Dame, Notre Dame, IN, USA, ORCID: 0000-0002-6051-0673
¶Software Sustainability Institute & ECS, University of Southampton, Southampton, UK, ORCID: 0000-0002-6809-5195
‖Software program, Australian Research Data Commons, Univ. of Technology, Sydney, Aus., ORCID: 0000-0001-9448-4023
**Berkeley Institute for Data Science, University of California, Berkeley, CA, USA, ORCID: 0000-0002-0233-1757
††Information School, University of Washington, Seattle, WA, USA, ORCID: 0000-0002-6008-3763

***Abstract*—Research software is essential to modern research, but it requires ongoing human effort to sustain: to continually adapt to changes in dependencies, to fix bugs, and to add new features. Software sustainability institutes, amongst others, develop, maintain, and disseminate best practices for research software sustainability, and build community around them. These practices can both reduce the amount of effort that is needed and create an environment where the effort is appreciated and rewarded. The UK SSI is such an institute, and the US URSSI and the Australian AuSSI are planning to become institutes, and this extended abstract discusses them and the strengths and weaknesses of this approach.**

***Index Terms*—software sustainability institutes, software best practices, community, research software, software sustainability**

## I. Introduction and context

A large fraction of modern research depends on research software [1, 2]. This software often can be considered to fit into a stack, as described by Hinsen [3], consisting of, from top to bottom: project-specific code (e.g., scripts, workflows), domain-specific tools (e.g., community codes), scientific software infrastructure (e.g., math and I/O libraries), and non-scientific software infrastructure (e.g., compilers, standard libraries), all on top of the operating system (OS) and hardware.

The fact that this software works at a given time does not mean that it will work in the future, because the lower layers upon which the software depends (including the OS and hardware) will change over time, and this can cause the software to break. Additionally, there may be bugs discovered in the software that should be fixed, and in many cases, new use cases will arise that require modifying the software. In all cases, there is work needed to keep the possibly fixed or expanded software running correctly on the possibly updated lower layers. We use sustainable to refer to situations where this work is planned for and likely to occur, though in some sense, sustainability is something that can only be evaluated in hindsight; sustainability in the future is merely a prediction.

While some of this software, particularly in the lower layers, is reasonably well-supported by funding agencies, industry, or open source communities, and is likely to be sustained, much of it is also developed by itinerant laborers (graduate students and postdocs) as a side effort in their research activities, and even when a person dedicates long term effort to a software package, at some point they will retire or move on.

## II. Challenges

Better sustaining of research software can be addressed in multiple ways, including by reducing the work needed to sustain it, and increasing the resources available to do so. To reduce the work, research is needed to develop, maintain, and disseminate best practices and effective tools. To increase the resources, which are often people, since developing and maintaining software is human-intensive, we must increase human effort, either by paying people to work on software or finding other ways to encourage them to choose to do it.

While there is a technical aspect to these two factors (e.g., determining best practices, developing tools), many issues are partially or entirely social (e.g., disseminating best practices and tools, incentivizing contributions to software via hiring and promotion, funding policies that plan for ongoing maintenance), thus community activity is needed to address them.

## III. Existing solutions

These challenges can be addressed at a global or national scale, and because research funding is typically national, national efforts are arising. Three national activities have been working on this problem in their own countries, as part of a larger community of researchers interested in this problem.

In the UK, the Software Sustainability Institute (SSI, software.ac.uk) was established in 2010 with EPSRC funding to identify key issues and best practices for research software [4]. It has worked with 70+ research groups to improve their software practices and published over 80 guides (some used by >50,000 people) ranging from software engineering to managing projects to building communities. Other successes include the SSI's Fellowship program, a cohort of 150 advocates for improving software practice (including diversity and recognition) in their research domains [5], and a partnership with The Carpentries that developed a 350+ instructor base and delivered training for 7,500 learners at 50+ organizations. Consequences include: an increased use of software engineering practices that improve sustainability, such as version control; software outputs from research projects are more likely to be developed as open source and deposited in a digital repository.

The SSI was instrumental in establishing the Research Software Engineering movement, from the term Research Software Engineer being coined at a 2012 SSI workshop [6] to providing *backbone organization* support to establish the UK RSE Association in 2014 and Society of Research Software Engineering in 2019. This included working with policymakers to develop funding programs such as RSE Fellowships [7] and guidance on recognizing software as a research output in the UK Research Evaluation Framework. There are now RSE associations on three continents, 370+ professional members, and thousands engaged in the wider RSE community. UK researchers are more likely to explicitly budget software engineering effort on research grants.

In the US, a planning project started in 2017 to design a US Research Software Sustainability Institute (URSSI, urssi.us) and to build community support for it [8]. The project has run multiple workshops and a large survey to understand US researchers' needs and to determine possible activities to support them. In addition, it has used a set of community activities (e.g., website, blog posts) to build community awareness. These activities and examining the UK SSI's results led to a plan for a US institute: plan.urssi.us. This plan includes work in community and outreach, education and training, incubation, and policy. The URSSI team is now seeking funding to turn its vision, or at least parts of it, into reality.

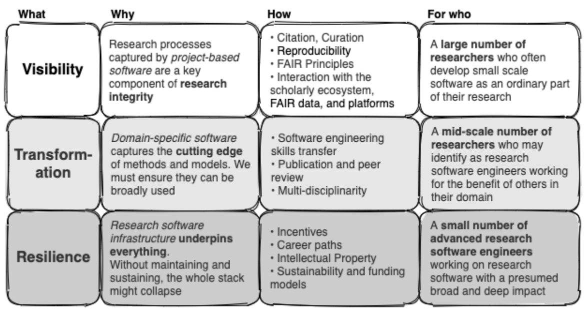
In Australia, work began in 2020 to draft a national agenda for research software (agenda framework shown in Figure 1), with the aim of achieving recognition of software as a first class research output. This agenda is heavily informed by the work of the UK SSI and URSSI. Establishing a software sustainability institute, AuSSI, will be validated with the community in 2021. AuSSI will serve to coordinate activities arising from the agenda, and will be housed within, and complement existing activities at, the Australian Research Data Commons (ARDC). The agenda suggests activities in infrastructure, guidance and outreach, community building and advocacy.

## IV. Strengths and Weaknesses

As a solution to making research software more sustainable, institutes have some strengths:

- Focus on academia and national labs (a government funded activity can be most successful aimed at government-funded institutions)
- Collaboration between institutes and communities (institutes and individuals can work together under a common vision to find common or customizable solutions)
- Leveraging research-specific concerns such as reproducibility and open science / open research

and some weaknesses:



| What | Why | How | For who |
|---|---|---|---|
| **Visibility** | Research processes captured by *project-based software* are a key component of **research integrity** | • Citation, Curation<br>• **Reproducibility**<br>• FAIR Principles<br>• Interaction with the scholarly ecosystem, **FAIR data, and platforms** | A **large number of researchers** who often develop small scale software as an ordinary part of their research |
| **Transform-ation** | *Domain-specific software* captures the **cutting edge** of methods and models. We must ensure they can be broadly used | • Software engineering skills transfer<br>• Publication and peer review<br>• Multi-disciplinarity | A **mid-scale number of researchers** who may identify as research software engineers working for the benefit of others in their domain |
| **Resilience** | *Research software infrastructure* **underpins everything.** Without maintaining and sustaining, the whole stack might collapse | • Incentives<br>• Career paths<br>• Intellectual Property<br>• Sustainability and funding models | A **small number of advanced research software engineers** working on research software with a presumed broad and deep impact |

Fig. 1. Framework for AuSSI agenda. (FAIR = findable, accessible, interoperable, reusable)

- Hard to impact industry (industry is much larger and has different social practices and incentives)
- Limited funding and limited time (government funded activities are dependent on government funding, which is generally of fixed duration)

Institutes are not a complete solution, but in cooperation with other activities (and with each other), they can help communities change practices, leading to more sustainable research software.

## References

[1] S. Hettrick *et al.*, "UK research software survey 2014," Dec. 2014. [Online]. Available: https://doi.org/10.5281/zenodo.14809

[2] U. Nangia and D. S. Katz, "Understanding software in research: Initial results from examining Nature and a call for collaboration," in *13th Int. Conf. on e-Science*, 2017. doi: 10.1109/eScience.2017.78 pp. 486–487.

[3] K. Hinsen, "Dealing with software collapse," *Computing in Science & Engineering*, vol. 21, no. 3, pp. 104–108, 2019. doi: 10.1109/MCSE.2019.2900945

[4] S. Crouch *et al.*, "The software sustainability institute: Changing research software attitudes and practices," *Computing in Science & Engineering*, vol. 15, no. 6, pp. 74–80, Nov. 2013. doi: 10.1109/mcse.2013.133

[5] S. Sufi and C. Jay, "Raising the status of software in research: A survey-based evaluation of the software sustainability institute fellowship programme," *F1000Research*, vol. 7, p. 1599, Oct 2018. doi: 10.12688/f1000research.16231.1

[6] R. Baxter, N. Chue Hong, D. Gorissen *et al.*, "The Research Software Engineer," Sep. 2012. [Online]. Available: https://www.research.ed.ac.uk/portal/files/65195747/DR2012_12_1_.pdf

[7] UKRI, "Research software engineer fellowships 2020," 2020. [Online]. Available: https://www.ukri.org/opportunity/research-software-engineer-fellowships-2020/

[8] J. C. Carver, S. Gesing, D. S. Katz, K. Ram, and N. Weber, "Conceptualization of a US research software sustainability institute (URSSI)," *Computing in Science & Engineering*, vol. 20, no. 3, pp. 4–9, 2018. doi: 10.1109/MCSE.2018.03221924