The Software Sustainability Institute: changing research software attitudes and practices

www.software.ac.uk

By Stephen Crouch, Neil Chue Hong, Simon Hettrick, Mike Jackson, Aleksandra Pawlik, Shoaib Sufi, Les Carr, David De Roure, Carole Goble, and Mark Parsons

To effect change, the Software Sustainability Institute works with researchers, developers, funders, and infrastructure providers to identify and address key issues with research software.

Software is critical to research. A recent survey showed that 84 percent of researchers view developing software as "important or very important for their own research."[1] Despite this exalted position, little emphasis is placed on developing good software, which meets the same rigorous specifications that researchers expect of their other tools.

Many researchers are yet to be convinced of the importance of developing well-engineered software. Although we might disagree with this viewpoint, it's an understandable one, because the research community provides little - if any - reward for producing such software. A lack of reward leads researchers to choose quick fixes over a more considered, maintainable approach to development. The effect is to burden much research software with an intractable technical debt (where work is needed to correctly engineer the quick fixes that have been implemented).

Researchers who are keen to develop or incorporate software into their research also face obstacles. It's difficult to gain the necessary development skills, because training in software engineering is difficult to obtain, and it's difficult to employ already trained developers on an academic project. In 2010, the Software Sustainability Institute was founded by the UK's Engineering and Physical Sciences Research Council (the largest of the UK government's seven research funding bodies).

The Software Sustainability Institute is a partnership between the universities of Edinburgh, Manchester, Oxford, and Southampton. Its goal is to cultivate world-class research with software, by overcoming the problems that beset research software and changing the way that researchers view it.

Reproducibility: A Study of Complexity

If we're to change the way that researchers deal with software, work is required on many fronts: on software development, on the community that uses and develops software, on the training that's available, and on influencing the policy that motivates all of the stakeholders in the research software community. To understand the complexity of these problems, we need only look at the issues around reproducibility.

Reproducibility is a core principle of science. As such, most researchers will have bemoaned the state of a graduate student's lab book, and argued for a clear, understandable, and transparent description of the methods and techniques employed in an experiment. Despite this familiar scenario, there's a conspicuous absence of these arguments being applied to software.

This isn't just a problem of researchers lacking the software engineering skills and motivation to test their software. The nature of software and the way it's developed makes it uniquely susceptible to fail a test of reproducibility. Not the least of these factors is that software is long-lived, providing ample time to incorporate errors or lose know-how, or indeed to lose the staff who are trained to use the software. In fact, the lifetime of software has increased to the extent that it often exceeds that of the hardware it runs on:

"Traditional trends are turning upside down. Hardware (the traditional capital investment) is becoming obsolete every two-to-three years, while software is becoming the capital investment. The standard scientific practice of constantly reinventing and rebuilding existing tools is becoming increasingly untenable."[2]

Software development practices compound the problem further. The culture of isolated development leads individuals (or individual groups) to overestimate the ease with which their software can be used, the transparency of the code, and even the possibility of locating the "right" version of the software (if, indeed, it still exists). Without the correct know-how, and with no documentation to acquire that know-how, the software can't be reused, and there can be no reproducibility if it's impossible to reuse the code.

As we can see with this discussion on reproducibility, the issues faced by research software are multifaceted. Our analysis is that the solution must also be multifaceted, which led us to split the institute into four teams, each of which would work to effect change on a different aspect of the problem.

Our consultancy team provides skilled software developers who work closely with researchers to help them develop better software and make better use of the supporting infrastructure. To learn from the research community about the software they use and the problems they face, our community team holds events and runs programs to fund researchers in exchange for intelligence. Then our training team uses the intelligence we gather to develop training that provides the skills that researchers require. None of our goals can be long-lived without cultural change in academia, so our policy team works to change the way research software is dealt with by all of the stakeholders in the research software community.

Helping Researchers Build Better Software

As software develops and matures, it reaches barriers that prevent further improvement, growth, and adoption. These boundaries are caused by a lack of skills in the group that developed the software, or by a lack of effort to focus on overcoming these boundaries. Technical debt is often a factor, since indebted software becomes unmaintainable and unable to evolve.[3] The institute's consultancy team works with researchers to address the barriers their software has encountered and preempt future barriers.

The consultancy team runs an "Open Call" for projects, which is a regular competition that allows researchers to make the case for help improving their software. Resources are limited, so the perennial problem is that of finding the right projects to work with, which is a decision we make based on the potential impact of the project (its importance to the research community), as well as the match to skills in the consultancy team and the commitment of the researchers to work with us.

Working across the UK research community, the institute provides consultancy to projects from a variety of research fields (see Figure 1).

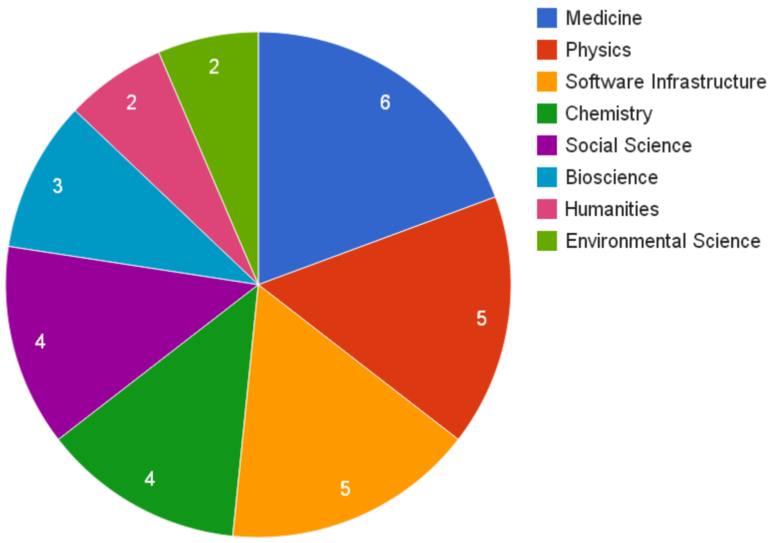

Figure 1. Institute consultancy projects per research field

The Adaptive Multi-Resolution Massively Multicore Hybrid Dynamics (AMRMMHD) project is a good example of the kind of work completed by the consultancy team. The project helped turn a one-man, small-scale software project into a successful multideveloper program that's transforming research in molecular binding.

The SIRE software that formed the basis of the project is used to predict whether molecules will stick together: whether a drug will stick to a protein, for example. It had been developed by a lone chemistry researcher - Christopher Woods - who wanted to bring more developers into the project, which required a cultural change within the project. As Woods described, "The code by this stage was huge, and I knew every line. To let someone else come in, without constantly reviewing their code, was really hard."

The consultancy team worked with Woods to understand the code's structure and the way in which the AMRMMHD project operated, developing a plan for managing the code that helped to separate and manage new software development. Ultimately, the consultancy team taught Woods how to be a software manager rather than a developer.

In addition to working on project management, the consultancy team reviewed the implementation of improved dynamics sampling algorithms used in the software, which has enabled these to run on a wider range of computer resources, including the BlueCrystal supercomputer and Emerald GPU cluster in the UK.

The institute is also working with the ForestGrowth-SRC (Short Rotation Coppice) project to assist its exploitation of institutional HPC infrastructure, and increase the reusability of its modeling software. ForestGrowth-SRC is a model for woody biomass yield prediction from an area of land, and takes into account the hundreds of different biological processes that contribute to yield, such as photosynthesis, light interception, and the way the plant uses carbon. Currently, scaling up to 100,000 square kilometers or more (the area of the UK), with ranges of starting inputs for temperature, precipitation, and carbon dioxide levels means that a run takes many days to complete. To improve these simulation run times, the Fortran Windows code has been ported to Linux to enable simulations to exploit the University of Southampton's Iridis4 cluster. This has the added benefit of increasing the code's reusability in other projects that commonly use different platforms, an important goal for future collaborations.

We're also assisting the developers of the Imperial College Lower Limb Model to move towards open development and reduce their software's technical debt. Their modeling software is used to predict muscle and joint forces at the hip, knee, and ankle by employing motion data obtained through gait analysis. Following a review of their codebase and development practices, we're now assisting them to share their code via an online revision control system (rather than sharing USB drives), and teaching them the skills required to use this in their development work. This work will position the team and the software to exploit other potentially lucrative research areas where there's collaborative interest, such as cancer and surgery, and to open source their software at a future date.

By helping software projects such as those above to directly reduce software technical debt, adopt best software development practices and infrastructure and expand their community of developers, the consultancy team assists the research community to ensure a sustainable future for their software. In doing this work, we assist projects in improving software reusability, a critical and necessary prerequisite for reproducibility.

Addressing the Skills Gap: Training

Researchers are required more frequently to write or modify software. Unfortunately, researchers often receive little formal training in programming or software development, instead picking up skills in an ad hoc, self-guided manner. This lack of sufficient skills risks rapid accumulation of technical debt in software, and has negative consequences for reuse and reproducibility. As well as providing a set of comprehensive practical guides on a variety of software-related topics, the institute's training team provides a number of training services to address this skills gap.

The most significant of our training activities is Software Carpentry (www.software-carpentry.org), an international initiative to give scientists and researchers the software development skills they need to achieve more in less time, and with less difficulty. Software Carpentry's primary vehicle is the "boot camp," an intensive, highly interactive two-day course in software development skills and how these contribute to research's best practices. The institute has made a substantial contribution to Software Carpentry since 2011, instructing at a total of 14 boot camps worldwide to more than 500 researchers, and it has organized, and helped others organize, 10 further boot camps. Since February 2013, we have coordinated Software Carpentry within the UK.

Software Carpentry is an effective vehicle in addressing the software development skills gap. An independent assessment of Software Carpentry carried out in the US over the first six months of 2012 concluded that Software Carpentry "increases participants' computational understanding," "enhances their habits and routines," and "leads them to adopt tools and techniques that are considered standard practice in the software industry."[4]

One of our goals is to help Software Carpentry become self-sustaining in the UK, to ensure the high demand for Software Carpentry continues to be satisfied and the skills gap continues to be addressed. To this end, we encourage researchers to become future helpers and instructors, and since February 2013 we've helped four researchers to become helpers and four to become instructors. In 2013, the institute also offered Software Carpentry training packages as an EPSRC-approved training provider, through the EPSRC call for Centers for Doctoral Training (CDTs). We'll deliver our training package and provide advice on how to run boot camps to a total of 10 consortia within the UK over the coming year. This activity has had impact within EPSRC, promoting the importance of developing software skills for research.

We're also helping to address a skills gap in the use of HPC resources. Collaborating with Software Carpentry and the Distributed Research utilizing Advanced Computing (DiRAC) consortium, we've developed a "driver's test" for the UK DiRAC-integrated supercomputing facility. This test not only represents an aptitude gateway to the resource, but serves to encourage researchers to undertake self-guided training to improve their software development and research skills. The driver's test is now being rolled out across seven regional sites, to run through 240 users, including postdoctoral research associates and PhDs by the end of this year.

The institute also delivers a number of other smaller-scale training services in the UK, engaging with other training initiatives, collaborations, and institutions to deliver talks and workshops on software development best practices. This includes running workshops on software sustainability, chairing discussions on "what makes good code good," and organizing meet-the-expert sessions. To date, we've run 12 sessions to more than 400 researchers across the UK.

There's evidence that many researchers want to improve their skills. In a survey of more than 400 respondents in the domain of species distribution modeling, 79 percent of scientists expressed a desire to learn additional software and programming skills[5]. This is indicative of a wider trend in research, which is reflected in the significant demand for further Software Carpentry boot camps - the institute is planning a further seven in the UK in the next six months alone. Through its training activities, the institute continues to address this software development skills gap. Helping researchers to resolve - and more importantly, preempt - the accumulation of technical debt in the software they develop is a key step to enhance research software reuse and promote reproducibility.

Supporting the Research Software Community

The UK research community faces a number of issues associated with software reuse, reproducibility, and accumulation of software technical debt. To understand how to address these and other issues, the institute must first build an understanding of them from the research community. The institute's community team is focused on gathering intelligence on these issues and many other aspects of research software from the community, allowing us to find solutions and develop policy to influence positive change.

The institute's Fellowship Program is a cost-effective approach to obtain community intelligence, recruiting members of the software research community in a way that benefits them and the institute. The Program provides bursaries to researchers across many domains, institutions, and career stage in exchange for their expertise and advice. Institute Fellows receive funding for attending conferences and workshops that focus on different aspects of software development and use in their research area. In exchange, they supply domain intelligence that we use to develop wider policy objectives. As domain specialists, they're also ideally placed to identify other institute opportunities (such as consultancy projects) and they also provide feedback on activities undertaken by the institute. In 2013, the institute funded 15 Fellows from a range of different research areas, from history to magnetic resonance imaging (MRI), and is putting in place another Fellowship Program for 2014.

Given the limited budget, the Fellowship Program represents a significant return on investment, with the Fellows having established a keen group identity, vision, and set of activities that assist the UK research community in many areas. A remarkable outcome of the program is that many Fellows go beyond their initial remit, taking initiative in organizing events within their respective communities, delivering institute-related presentations, advocating best practices, and suggesting other activities that have relevance and benefit to the institute. A major event being organized and attended by four Fellows is a "Software & Research Town Hall Meeting" at the American Geophysical Union (AGU) Fall Meeting; the AGU is one of the largest community events, it's attended by 22,000 researchers from all over the world.

Fellows have also led events that have focused on addressing the software development skills gap and discussing cultural issues. For example, in coordination with the training team, our Fellows have organized three Software Carpentry bootcamps and instructed in two of them, as well as organizing a practical Web Scraping for Arts, Humanities, and Social Sciences Workshop (www.software.ac.uk/web-scraping-arts-humanities-and-social-sciences-workshop-2013). On a cultural level, the Fellow-led Software in Polar Research Workshop (http://polarnetwork.org/events-and-workshops/2013-software-and-polar-research-workshop) included discussions on the prevalent culture of reinvention over reuse.

At a wider level, as a service to the community and also as a means to gather further intelligence, the institute also holds an annual Collaborations Workshop that brings together different stakeholders of research software - including researchers, software developers, managers, and funders. The goals are to explore important ideas, concerns, and issues in software, and to help participants forge interdisciplinary collaborations. The Collaborations Workshop is run atypically (not in a conference style), thereby empowering the participants to propose and discuss topics of their own choice and interest during the breakout sessions. Lightning talks provide a great opportunity to present software tools, research projects, ideas, and events.

The institute represents the entire UK research community involved in the use, support, and development of software in research, working alongside domain-specific activities such as Sound Software (for audio and music research) and the Collaborative Computational Projects, which encourage widespread and long-term use of high-end scientific codes. This is clearly a huge undertaking, so the institute focuses on scalable activities that present a high return on investment, such as the Fellows Program and the Collaborations Workshop. The community intelligence that has been accumulated across a wide variety of topics is considerable (http://software.ac.uk/resources/conference-intelligence), and has led to a number of policy initiatives that are having an impact within the research community.

Improving Software's Status and Beyond

The status of software in research won't improve without changes of opinion at all levels in the research community. This herculean task is the job of the institute's policy team.

To create our manifesto, we reviewed the changes we would like to see, based on our experience and on feedback from our community. This led to four policy areas:

- We'll promote software as a research output, so that researchers get recognition for developing good software.
- We'll promote and provide software training for researchers, to increase the skills base of the research community.
- We'll campaign for research software to meet the same reproducibility standards as other research tools, which will help to increase software's standing within the research community.
- We'll campaign for recognition and reward for research software engineers, a hidden yet fundamental group to the development of research software, which is also a good case study of the policy teams' work.

Research software engineers (RSEs) are the people behind research software. The fundamental difference between an RSE and any other software developer, is that RSEs understand both software and the research that it makes possible. They generally start out as researchers and then get interested in software, or start as software developers who get interested in research. Without RSEs, most research software wouldn't exist, but despite this fact, the role often goes unrecognized and unrewarded in academia.

The rise of the RSE is a relatively new phenomenon, so it's understandable that academia is taking some time to adapt. The policy team is working to increase the pace of change, by publicizing the importance of the RSE role (http://www.timeshighereducation.co.uk/news/save-your-work-give-software-engineers-a-career-track/2006431.article), by gathering statistics on RSEs and their work, and by bringing RSEs together to form a community.

RSEs can be viewed as "super users" of software in research. They're the people who other researchers consult when they're having problems with their software. As such, RSEs feed information into the research community, making them the perfect people to disseminate advice on good software.

Our early work with RSEs has shown that they're ahead of the curve when it comes to understanding the issues that affect research software. We've found that the community shares our concerns about reusability, reproducibility, and the lack of software skills. We've found that RSEs are keen to partner with the institute to change not only their own status in the research community, but also that of software.

The next step is to gather statistics about the size and composition of the RSE community and use this evidence to petition research stakeholders - and universities in particular - and persuade them to recognize the substantial contribution RSEs have made to research.

In addition to the RSE campaign, the policy team is also working to bring together e-Infrastructure training providers. So that they provide more uniform and synchronized training across the UK, we're working on a campaign to highlight the difficulty of understanding institutional approaches to intellectual property law concerning software developed in research.

The policy team is working to overcome issues that require significant changes in the way the research community deals with software and the people who are responsible for that software. These are not issues that will change overnight, but the institute is perfectly placed to help accelerate the pace of change using the expertise and contacts developed by all of the institute's teams.

The Need for International Initiatives

While the remit of the Software Sustainability Institute is to support UK researchers, the need for worldwide research software support initiatives requires national and international coordination. Research collaborations often involve groups working across several countries, and the software that supports this research will be chosen by these consortia. The institute is aware of and provides advice to many other initiatives. In the US, the National Science Foundation (NSF)-funded Software Institutes for Sustained Innovation (SI2) program has funded groups to work on specific codes, and more widely to support particular disciplines, such as the water sciences, earth sciences, and computational chemistry; as well as cross-cutting concerns, such as cybersecurity and trust, science gateways, and new accelerator platforms. The Sloan Foundation is supporting international initiatives to improve the use of software in research and address the skills gap, including the aforementioned Software Carpentry, rOpenSci (for scientists using R software) and ImpactStory (which provides alternative metrics of impact that include software production). In Europe, the European Commission has hinted it will support Centers of Excellence in scientific software in the new Horizon2020 framework program, and many national funders are considering how to provide additional funding for software maintenance.

What's clear is that in many disciplines, the critical mass of expertise and development effort can only be achieved if the community of users consolidates on a few community codes. This requires both international coordination and cooperation, but also the ability of national funders to support software that has been developed elsewhere. The institute's consultancy work has pioneered this model, helping codes like CP2K (originally developed in Switzerland) that have a large UK user base. This is further being trialed via the joint NSF and Engineering and Physical Sciences Research Council (EPSRC)-funded computational chemistry projects, including one in which the institute is helping to bring together several US and UK codes and models, including Assisted Model Building with Energy Refinement (AMBER), Atomic Multipole Optimized Energetics for Biomolecular Applications (AMOEBA), Open Molecular Mechanics (OpenMM), and TINKER.

Software permeates all disciplines and all stages of the research process, from observation to publication. The institute's remit is huge: supporting this software across all disciplines. To effect the necessary changes, the institute focuses its effort on different aspects of the research community through its work on consultancy, community, training, and policy. Ultimately, our work isn't just about improving a number of software packages, it's about changing cultural attitudes and practices towards software in general. While our consultancy work directly addresses issues with software, it also embeds good software practice into a research group. Our community team generates contacts across the research community, and programs like our Fellowship bring new disciplines into the debate about research software. The training team directly addresses the lack of software skills in the research community, and viral training like Software Carpentry ensures that today's trainees are tomorrow's trainers. Our policy team draws together the findings of everyone in the institute and campaigns for a culture change that will benefit all stakeholders in the research community.

Researchers are increasingly recognizing that software is the foundation to their research, but there are several challenges to overcome before software gains the status it deserves. Recognizing software as a research output will bring recognition and reward for software, and make it credible for researchers to focus their efforts on development. This in turn will ensure that researchers will plan for the long-term, which should reduce technical debt and - ultimately - lead to the development of reproducible software. At the same time, a greater understanding of good software practices will lead to more robust and reliable software, which will increase confidence and lead to greater software reuse.

Acknowledgments

The authors would like to thank Malcolm Atkinson, a previous co-investigator, and Rob Baxter, the institute's former development manager, for their contributions to the project, as well as other members of the SSI Team for their input. The Software Sustainability Institute is supported by the Engineering and Physical Sciences Research Council (ESPRC) grant EP/H043160/1. Funding for specific projects has been provided by the Biotechnology and Biological Sciences Research Council, ESPRC, and Jisc.

References

1. J.E. Hannay et al., "How Do Scientists Develop and Use Scientific Software?" Proc. 2009 ICSE Workshop on Software Eng. for Computational Science and Eng., IEEE CS, 2009, pp. 1–8.
2. D. Atkins et al., RCUK Review of e-Science: Building a UK Foundation for the Transformative Enhancement of Research and Innovation, tech. report, Eng. and Physical Sciences Research Council, 2010; www.epsrc.ac.uk/SiteCollectionDocuments/Publications/reports/RCUKe-ScienceReview Report.pdf.
3. N. Brown et al., "Managing Technical Debt in Software-Reliant Systems," Proc. FSE/SDP Workshop on Future of Software Eng. Research, ACM, 2010, pp. 47–52.
4. J. Aranda, Software Carpentry Assessment Report, tech. report, 2012; https://github.com/swcarpentry/assets/blob/master/assessment/5.0/aranda-assessment -report-2012-07.pdf.
5. L.N. Joppa et al., "Troubling Trends in Scientific Software Use," Science, vol. 340, no. 6134, 2013, pp. 814–815. DOI: 10.1126/science.1231535