

Empowering research

The value of software

The value of software is frequently overlooked. Software is valuable both as a tool for researchers, and as an output from research projects - and in this role it is also associated with a cost to the research councils. Researchers can increase the value extracted from software, but only if changes are made to the way that software is managed. This should begin with an appreciation of software's real value.

This briefing paper has been written for Research Council staff. It describes why we believe that long-term software sustainability and preservation plays a vital role in empowering high-quality research. It also highlights ways in which we think the Research Councils could help to improve practice in this area.

The suggestions in this briefing paper are based on work performed by the (EPSRC- and JISC-funded) Software Sustainability Institute, in partnership with Curtis+Cartwright Consulting Limited, and builds on previous work undertaken at the STFC.

Why is software sustainability important?

Software is expensive to develop and maintain, yet much software is not used at its full capacity, and some even reproduces functionality available elsewhere. Resources can be saved by encouraging the wider use, and sharing, of useful and reliable software. This practice not only saves money, it can also provide a permanent record of research and is an opportunity for improving the quality of research.

The sustainability of software is an emerging area of interest for the Research Councils (e.g. the EPSRC recently funded a tranche of software-sustainability projects). This interest can only increase as the financial climate leads to more pressure on budgets. Improving the efficiency of software use through sustainability could help reach spending targets. Much needs to happen before sustainability becomes commonplace, including changes at all levels in the research community.

The RCUK Review of e-Science 2009 said that "Software development is not basic research, but instead requires a critical mass of full-time engineering professionals, paid market wages and supported along a genuine career path. There needs to be a commitment to long-term funding and a recognition that software development and support is at least as important to modern science as massive accelerators and telescopes."

What are the benefits?

Lower research costs

Software sustainability leads to software re-use; re-use prevents duplication. Developing robust, production-quality software is very expensive. Minimising duplication can significantly reduce the cost of developing new software, whilst boosting quality and reliability.

Better quality research

Continued access to software, data and services means that research results can be reproduced and repeated. This reduces both unintentional errors (because of increased cross-checking) and deliberate research fraud. When software is sustained, it can be used to re-analyse old data in the light of new theories. Old data can also be combined with new data, which can offer new insight and knowledge. Sharing of software also leads to sharing of ideas, creating collaborations that lead to better research.

Increased research impact

When software is sustained and preserved, the original investment offers a return over a longer time frame. Some re-use will be planned and foreseen, but some will be serendipitous, leading to potentially important research discoveries.

What are the issues?

There are excellent examples of long-lived software, which form the bedrock of entire research communities. These examples are very much in the minority, since much software is kept private and is considered disposable. We have identified key two reasons for a lack of software sustainability and preservation in the research community.



1. It is difficult to gain funding for sustainability

Software does not intrinsically have a long shelf life. Without careful maintenance, software quickly becomes unusable – a process known as **software decay**. However, funding is more readily available for developing new software than for maintaining existing software. New software is developed rather than established software being sustained. This is a significant waste of resources.

2. Researchers face disincentives and uncertain returns if they share code

Most researchers aren't software developers, and few have the time to learn software engineering. A researcher's own software is generally *no frills* software that is coded quickly to perform a specific task. To share this code, researchers would have to invest time into cleaning up their code and supporting new users, which is not their job. The return from sharing code can be uncertain: a low uptake would bring into question the effort invested into preparing the code for sharing, and there is no easy way of citing software, which means that researchers receive no credit for their software.

What can the Research Councils do?

The risks of not sustaining and preserving data are well understood. The same risks apply to software too. This was recently highlighted in the *Climategate* reviews, which referred to algorithms and code in the same manner as the data itself.

Where there are policies for data preservation, there should be policies for software preservation and sustainability. Our engagement with researchers has highlighted the need for policies that:

- encourage long-term software retention, access, sharing and citation
- discourage duplicative software development in research grants
- provide specialist support in software sustainability since researchers often cannot, and do not want to, sustain software themselves

- encourage improved software engineering practice by researchers, particularly training and professional development for PhD students and researchers early in their career
- encourage a managed software development lifecycle with appropriate decisions and funding at each stage. Not all software will move to more and more mature stages, but tailored funding in appropriate proportions should be available at each stage (e.g. prototype to research quality code, or continuing development of research quality code). Generally there is a shortfall for continuing development.
- encourage those researchers who are software developers to take a lead role in building further capacity and in making strategic decisions in software sustainability so the right software components are selected for continuing development

Four more things you should know

Does all software need to be sustained or preserved?

No. Being selective is a smart strategy.

Is this only a UK problem?

No. Issues around software sustainability are common in the US and elsewhere too, especially since research repeatability and reproducibility has become an important topic.

Whose responsibility is this?

There is no single responsible party. Our project's stakeholders were adamant that a combination of top-down interventions from the Research Councils and bottom-up improvements from software developers is necessary.

Is open-source software the answer?

Open development and open licensing are good practice, but in themselves cannot guarantee sustainability or preservation.

Further information and useful resources

The Software Sustainability Institute

<http://www.software.ac.uk>

How do scientists develop and use scientific software?

<http://bit.ly/VgWWW1>

The Scientific Method in Practice: Reproducibility in the Computational Sciences

<http://bit.ly/cKgCGu>

Blue Ribbon Task Force on Sustainable Digital Preservation and Access final report

<http://brtf.sdsc.edu>