



Your project is ending...

Don't panic!

It's a sad time. Your project is finally ending. You've broken barriers, realised new possibilities, and pushed the envelope. Or at least you've nudged it a bit. All you need to do is close shop, turn off the lights and sit back in a comfy armchair... but wait! What about the software you developed?

The software that discovered the origin of the universe, stopped global warming and achieved cold fusion – the one that led to three Nobel Prizes and 450 publications. Is it possible that someone might just have a use for it? Perhaps it should be saved from a fate worse than deprecation! But how do you decide? And how should you save it? Well firstly... don't panic!

Whether for research, administration, learning or teaching, software is an increasingly valuable research tool and output, and needs to be managed as such. This is a briefing paper for researchers, software developers and managers who are coming to the end of a project and need to decide what to do with software that has been developed. This paper will help you make decisions about what to do at the end of a project by outlining the options and providing guidance.

To ensure valuable ideas and investment are not lost, decisions need to be made on what, if anything, is to be done with the software and research outputs from the development process. Sometimes the software you developed will be easily reproducible (e.g. it's based on a published specification or algorithm), or was just a useful training exercise that need not be sustained. At other times, you have the opportunity or obligation to sustain or preserve your software for longer. Just because you've finished with it now, doesn't mean you or someone else won't want to return to it at a later date.

Why is software sustainability important?

A lot of time and effort goes into developing software, and into other outputs from the development process such as design documents, code and binaries. However, the project-based nature of research funding makes it common for researchers to move quickly from one topic to another, with the result that software is often

seen as disposable. Come the end of a project, it can be easy to contribute to the growing amount of *abandonware* without really considering whether that is the right choice.

If software is sustained, it can have a far greater impact. Unfortunately, it is often only at the end of a project that the utility and potential future of the software becomes clear.

Should your software be sustained?

The first thing step is to decide whether your software is needed by others, or if there is anything further you would like to do with it. We have identified four purposes for taking further action

1. Encourage software reuse

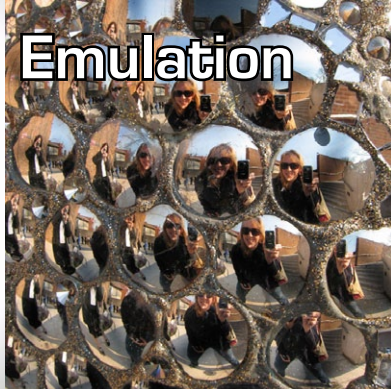
Does your software have the potential to be used by others?

2. Achieve legal compliance and accountability

Does your software need to be preserved for audit, integrity, regulatory or general public accountability reasons?

3. Create heritage value

Does your software have intrinsic heritage value?



Cultivation

Keep your software *alive* by moving to a more open development model, bringing on board additional contributors and spreading the knowledge of processes. Start by gauging the scope for developer and user communities to see if this is feasible.

Further information and useful resources

Software Sustainability Institute – further resources for software sustainability and preservation, including a crib sheet of usage scenarios and a detailed benefits framework

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

OSS Watch – expert and FE/HE-specific resources covering a wide range of issues relating to open source software use and development

<http://www.oss-watch.ac.uk>

The Significant Properties of Software project – a JISC funded study into what properties are needed to allow software to be systematically preserved

<http://bit.ly/eF7yNv>

4. Enable continued access to data and services

Does preserving your software mean that you or other users can continue to use data and services that would otherwise be lost?

4. Taking action

Our detailed guidance (see *Further information*) contains lists of various scenarios where it is advisable to sustain your software, and the corresponding benefits in doing so.

When you come to the end of your project, there are three four options based on the need to sustain your software.

Technical Preservation

Preserve the original hardware and software in its current state. Start by determining the expected lifetime of each hardware and software component and whether purchasing hardware and media spares is necessary.

Emulation

Emulate the original hardware and operating environment, whilst keeping the software in its current state. Start by seeing if a suitable emulator already exists.

Migration

Update your software as required to maintain same functionality, porting/transferring before platform obsolescence. Start by reviewing the portability of the software.