www.software.ac.uk www.curtiscartwright.co.uk

Sustainability and preservation framework

An investigation of software preservation has been carried out by Curtis+Cartwright Consulting Limited, in partnership with the Software Sustainability Institute (SSI), on behalf of the JISC. The aim of the study was to raise awareness and build capacity throughout the Further and Higher Education sector to engage with preservation issues as part of the process of software development.

Part of this study involved examining the purpose and benefits of employing preservation measures in relation to software, both at the development stage and retrospectively to legacy software. The study built on the JISC-funded Significant Properties of Software study that produced an excellent introduction and framework to software preservation.

This is condensed version of a framework document. It is designed to assist developer groups and their sponsoring bodies to understand and gauge the benefits or disbenefits of allocating effort to ensuring that preservation measures are built into software development processes, and actively preserving legacy software.

Purposes, benefits and scenarios

A key challenge in digital preservation is being able to articulate, and ideally prove, the need for preservation. A clear framework of purposes and benefits facilitates making the case for preservation.

The table on the following page of this briefing paper shows a range of scenarios for each purpose to give some illustrative examples of where the purpose and accompanying benefits might be relevant.

We recommend that these purposes and benefits be combined with preservation plans regarding data and hardware: digital preservation should be considered in an integrated manner. For example, media obsolescence and recovery is often as much a part of a software preservation project as a data preservation project.

Note that if at all possible, especially where the software is an enabler, it's advisable to turn a software-preservation problem into a data-preservation problem. These problems are invariably easier to handle.

Do we need to preserve our software?

We do not believe that there is a simple and universally applicable formula for determining if your software needs to be preserved, and how to go about preserving it. Instead, we present thought-provoking questions and a range of factors which should be taken into account.

Questions you need to answer

Is the software covered by a preservation policy / strategy?

Is there a clear purpose in preserving the software?

Is there a clear time period for preservation?

Do the predicted benefit(s) exceed the predicted cost(s)?

Is there motivation for preserving the software?

Is the necessary capability available?

Is the necessary capacity available?

How should your software be preserved?

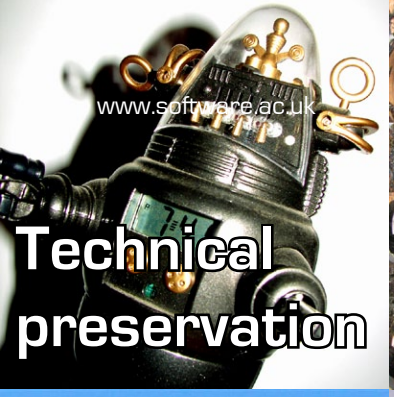
Seven different options for preservation and sustainability are presented.

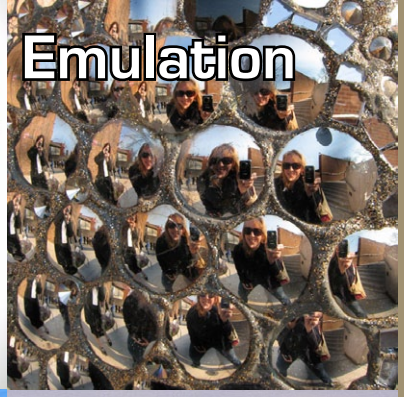
- **Technical preservation** (techno-centric) - Preserve original hardware and software in same state
- **Emulation** (data-centric) - Emulate original hardware and operating environment, keeping software in same state

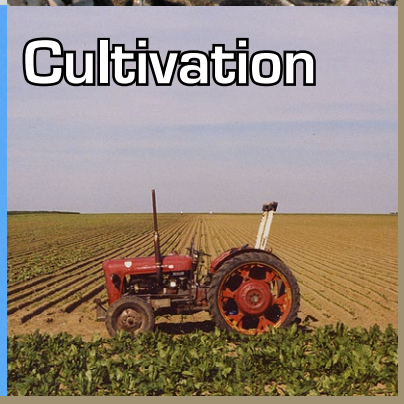
Purposes, benefits and scenarios

The table below shows a range of scenarios for each purpose to give some illustrative examples of where the purpose and accompanying benefits might be relevant.

Purpose	Benefits	Scenarios
Encourage software reuse	Reduced development cost Reduced development risk Accelerated development Increased quality and dependability Focused use of specialists Standards compliance Reduced duplication Learning from others Opportunities for commercialisation	Continuing operational use in institution Increasing uptake elsewhere Promoting good software
Achieve legal compliance and accountability	Reduced exposure to legal risks Avoidance of liability actions Easily demonstrable compliance lessens audit burden Improved institutional governance Enhanced reputation	Maintaining records or audit trail Demonstrating integrity and authenticity of data and systems Addressing specific contractual requirements Addressing specific regulatory requirements Resolving copyright or patent disputes Addressing the need to revert back to earlier versions due to IP settlements Publishing research openly for transparency Publishing research openly as a condition of funding
Create heritage value	(Heritage value is generally considered to be of intrinsic value)	Ensuring a complete record of research outputs where software is an intermediate or final output Preserving computing capabilities (software with or without hardware) that is considered to have intrinsic value Supporting the work of museums and archives
Enable continued access to data and services	**For research data and business intelligence:** Fewer unintentional errors due to increased scrutiny Reduced deliberate research fraud New insight and knowledge Increased assurance in results **For systems and services:** Current operations maintained Opportunity for improved operations via corrective maintenance Reduced vendor lock-in Improved disaster recovery response Increased organisational resilience Increased reliability	Reproducing and verifying research results Repeating and verifying research results (using the same or similar setup) Reanalysing data in the light of new theories Reusing data in combination with future data 'Squeezing' additional value from data Verifying data integrity Identifying new use cases from new questions Maintaining legacy systems (including hardware) Ensuring business continuity Avoiding software obsolescence Supporting forensics analysis (eg for security or data protection purposes) Tracking down errors in results arising from flawed analysis







- **Migration** (functionality-centric) - Update software as required to maintain same functionality, porting/transferring before platform obsolescence
- **Cultivation** (process-centric) - Keep software 'alive' by moving to more open development model bringing on board additional contributors and spreading knowledge of process
- **Hibernation** (knowledge-centric) - Preserve the knowledge of how to resuscitate/recreate the exact functionality of the software at a later date
- **Deprecation** - Formally retire the software without leaving the option of resuscitation/recreation
- **Procrastination** - Do nothing

Questions on how to preserve software

How much access do you have? (Owner, developer, access to source code, access to hardware, user)

Do you have the necessary Intellectual Property Rights (IPR)?

What are you needing to preserve? (A few major pieces of functionality, Most of the functionality, but tolerant of minor deviations, All functionality, but fixing errors when found, Must perform exactly as original)

What is your likely effort profile? (Something or nothing now, something or nothing in the future)

What is the maintainability of underlying hardware?

Is maintaining integrity and/or authenticity an important requirement?

How long do you want to preserve it for?

Can you afford it?

Are you also interested in further development or maintenance?

What development effort has been invested into the software so far?

Is the software open source? Could it be made open source?

Are there any barriers to making it open source?

Is the proposed approach appropriate to every purpose?

What are the relative advantages and disadvantages of each approach under consideration?

Further information and useful resources

The benefits framework document from which this briefing paper is derived, and many other sustainability resources can be found on the Software Sustainability Institute website:

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

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

curtis+cartwright

Software Sustainability Institute

JISC

This briefing paper has beevn preprared as part of a JISC-funded project by the Software Sustainability Institute in partnership with Curtis+Cartwright Consulting Ltd. It is part of a series of outputs that have been devweloped to support the HE sector by raising awareness of software sustainability and preservation issues in the software-development process. Front image courtesy of tarotastic. Back image courtesy of (clockwise from top left: Aussiegal, Lollyknit, Arnoooo and sarniebill1.