Document Distribution Note
Release to Members: Immediate
Release to Public: 07/05/2011

DigitalPreservationCoalition

Software Preservation Workshop – Curtis/Cartwright Consulting and Software Sustainability Institute

London, 7th Feb 2011

1. About the event

JISC has recently funded a report on the issues associated with the preservation of software, updating some original work first undertaken by STFC around 2008 and in anticipation of a slightly more diverse set of studies due in 2011. The workshop involved a relatively small number of participants and it examined the current state of the art, and future directions for preserving software. There are more details about the project and about the aims of the workshop online at: http://softwarepreservation.jiscinvolve.org/wp/2010/12/20/workshop-for-digital-curators/

2. Presentations and discussion

Kevin Ashley – Introduction

Much advice has focussed on data preservation, less on software. Indeed much of the advice about enabling independent utility of data is about reducing software dependence – such as advice to use TIFF or TXT files for long term data storage which are robust because it is reasonably easy to build viewers to render files. However, even with simple formats there is a considerable need to keep some representation information that will help make sense of even the simplest data standards: character set for example within TXT is not necessarily obvious – and at a higher level of meaning things like dictionaries are required to capture meanings of words. There are some scare stories of software preservation – for example one research project which ULCC worked with found there was a dependence on a simple component that changed the way a programme functioned producing a floating point error. It's usually true that errors produced by software can be harder to spot than errors in data. But if we understand software then it helps us understand the limits of what a researcher was able to do rather than what they got wrong. This may be particularly important when we need to put things to very different uses than originally intended. In any case, we know that files degrade in migration – most obviously in graphics which can quickly make a graphic – and it's obvious that there are problems with migration techniques. This may be specialist or rarely required but it will be an important use case: and its unavoidable.

Neil Chue Hong – Software Sustainability Institute

Software is not easy to define let alone preserve. Workflows for example will have individual elements of software and processes which are brought together as a batch that is greater than the sum of the parts. GIS is a case in point– a map output is actually produced from underlying raw data and is not really a thing in its own right. Which part is data and which part is process? As for video and sound, what parts are signal and signal processing, and which parts are data? Software has a whole technology stack and never stands alone: you can't really just preserve an operating system or create an emulator when software is a service and many components are disposable. Basic advice at

http://www.software.ac.uk/resources/approaches-software-sustainability includes different proposals for software preservation – technical preservation, emulation, migration, cultivation hibernation, deprecation and procrastination. Choice depends on different factors through time. Procrastination is not really an approach! Process centric or knowledge centric approaches will come be required for different use cases. (Q: Which is closest to what TIMBUS calls 'Exhumation'). Questions about software preservation come at a number of points in lifecycle of data and software – if you acquire data, if you are preparing to make data, if you are advising a third party about data you maintain. Key issues that emerge are rights management and costs of course, as well as technical challenges that you would expect. But useful elements to help you approach this systematically are significant properties of the software – key functionality for example. Software Sustainability Institute is wondering about a national facility for research software and improving software design and architecture, and embedding maturity models into software engineering. Good engineering should now include such thoughts: Report at: http://www.software.ac.uk/resources/preserving-software-resources

Brian Matthews – STFC

Software is not easy to define and there are multiple different ways to preserve it. Cultivation strategy is how STFC ensure that certain key software tools like ICAT are maintained. In simple terms, there's an active developer community that shares the source code and keep the source code alive and documented. The basic preservation steps for software are clear: preserve, retrieve, reconstruct and replay. Each of these sounds simple but is complicated in practice. For example a description of gross functionality will be required for retrieval, as well as general software architecture principle, provenance and licenses. Reconstruction is more complicated and we will need to provide an understanding of components and dependencies, programming language details including compilers and versions of compilers, operating systems, hardware, performance on specific machines and any auxiliary libraries or tools that are required to ensure a dependable response. Replay in turn will need documentation. The question arises whether you think you've done it well enough. Replay information might be considered as the significant properties and these in turn can be used to describe adequacy of a preservation service. The only way to ensure that this works is to provide sufficient test cases with measurable outputs. 'A calculation should be accurate to 8 decimal places' or 'The pagination should remain the same' or 'the system should respond on a specified time' or 'Roads should be rendered in red, not blue' for example are clear and offer specific measurable tests for different types of tolerance of adequacy of software preservation. Need to capture the significant properties early therefore!

Dialogue between Bram van der Werf and Steve Crouch – where do the needs of preservation and the developer meet or compete.

Build a house in the jungle and leave it, then it will be overgrown and rapidly destroyed. You need to maintain it, and when you live in it then there's nothing to worry about because you are continually maintaining it. Some software will be maintained in the simple case where there's a market to do so. There are some unusual cases where there is no market, but that's about users and their needs. Cultivation and open source are closely connected. Perversely, the more bug reports you receive

the more dependable your software is in the long term, because it means there's a community! All works very well for open source.

Group exercise: different perspectives on software preservation: who h as responsibility, who needs to be involved, what practical steps can be taken ...

- Need for advocacy before things will become critical
- Mix of needs, but even a migration path will create the need for some kind of access to software because preservation services can become obsolete
- Easier to maintain software or data in simulation
- Funders, developers, PI's need to take this on board

A concrete example where software preservation will be a realistic action

Medical imaging formats at the core which – and these are open standard. They are then analysed using a variety of tools and there are many different suppliers and the software comes from a variety of environments – some commercial, some free to use, some developed on demand. Libraries of resources may be commercially based and thus may not be completely open and may be proprietary. Life span of the data is >10 years and may be much more long lived.

3. Discussion points for DPC members

- Most data archives are not interested in migration and don't see the need to preserve software or are seeking to reduce their exposure to software preservation. But the components necessary to preserve data are themselves subject to obsolescence and consequently we may need to consider how to preserve and more effectively document preservation systems – made especially complicated the more that these are dependent on 3rd party services. In short, a migration strategy takes you to a place where it will be possible to limit your exposure to preservation of software but **migration pretty much guarantees that some software preservation will be necessary**.
- These issues are likely to become more important as distributed software services become more important and prevalent. For example, web archiving will surely have to develop some functionality to **archive apps** if it is to remain mainstream.
- Surprised that there was no discussion of registry services in this discussion nor event of the sort of interoperability framework which would make any sort of execution environment itself into a service. Imagine having a **software archive as service** available online on demand in some form.
- This is a research area and **it doesn't feel like it will be mainstream any time soon**. I'm not even sure where it sits on roadmaps like the PARSE.Insight roadmap. But it is close to the work funded for the TIMBUS project which to my mind adds an extra layer (perhaps of metadata) that makes sense of the business processes which will be called up through the software architecture. In any case, without a clear documentation of the business process which is being replayed much of the discussion will be academic and the use case unconvincing.
- Emulation and migration both have as their objective the rendering – by different means – of an information object that can be understood. OAIS refers explicitly to information

packages. Important to stress that what;s being described here is not so much an information package as an execution environment. This is **not about access to data so much as access to a process**. We might be interested in some of the old data too, but equally we may wish explicitly to use and process new data.

About this document

Version 1	Written on day	07/02/2011	WK
Version 2	Distributed	25/11/2010	DPC membership, Curtis Cartwright, SSI and TIMBUS partners