2006

An Open Source Approach to Medium-Term Data Archiving

Sherine Antoun
University of Wollongong, sma02@uow.edu.au

John Fulcher
University of Wollongong, john@uow.edu.au

Carole Alcock
University of Wollongong, carole.alcock@unisa.edu.au

S19th Bled eConference

eValues

Bled, Slovenia, June 5 - 7, 2006

An Open Source Approach to Medium-Term Data Archiving

Sherine Antoun, John Fulcher, Carole Alcock

University of Wollongong, Australia
sma02@uow.edu.au , john@uow.edu.au , carole.alcock@unisa.edu.au

Abstract

Medium- to long-term archiving of digital documents, beyond the lifespan of the authoring software/hardware, is a challenging problem. Magnetic and optical media are susceptible to environmental influences and deteriorate over time, often to the point where the archived documents can no longer be retrieved. Previous attempts to address this problem include migration and emulation, both of which have their attendant difficulties. It is the contention of the present study that an Open Source approach offers several advantages. More specifically, by archiving the Open Source application programs (in source code, not executable form) along with the documents in question, in both plain and compressed form, significantly increases the likelihood of being able to retrieve such archives at some future time. The application source code can be re-compiled to a form suitable for reading in (Open Source) viewers, thereby presenting to the user the archived document as the original author envisaged it. One set of experiments was undertaken distributing documents together with their (Open Source) authoring software via a Portable Virtual Machine (PVM) program to unused disk space on a network of SUN workstations. The success of this approach was evaluated using the following four measures: (i) lossiness of conversion, (ii) edit-ability, (iii) ability to save back to the original format, and (iv) functionality retention. Another series of experiments was conducted in which artificial ('speckle' or salt-and-pepper) noise was deliberately introduced to the archived documents in order to mimic degradation of the storage medium over time. It was found that survivability was heavily dependent on file type: simple text files and MPEG movies were impervious to even 18% introduced noise. Source code programs and JPEG images, by contrast, were intolerant to even the smallest noise levels (it has to be said however that straightforward re-editing of the former led to error-free compilation without much difficulty). Lastly, it was found that decompression (specifically the publicly available RAR decompressor) further enhanced the file recovery process. We conclude that an Open Source approach to the preservation of digital archives has considerable potential.

1. Introduction

Manuscripts from antiquity are still capable of being read in this day and age due, in large part, to the stability of the medium on which they were originally created: literally "etched in stone", in some cases. Preservation of records from more recent eras has proved more problematic: papyrus/paper records are susceptible to fire, flood and silverfish; early film stock was quite volatile: indeed, much was recycled for its silver content – and magnetic media decays over time (optical media less so). Modern media are considerably more fragile than those of earlier times. Archival storage of digital records is even more difficult, since bit loss can render the entire document unreadable. As Kahle (1996) rightly observes: "while it is possible to read 400 year old books printed by Gutenberg, it is often difficult to read a 15 year old disk". To cite another example: while the original lunar landing capsule (artefact) lives on in a NASA museum, its controlling software no longer exists (Open Channel Software, 2004).

Coupled with epoch shifts of media types has been the increase in mass availability of documents. At no other time has this been more the case than in this information age. Other characteristics of digital media are its ever-increasing storage capacity, its changing formats and the obsolete nature of authoring software (Rothenberg, 1999). All of this renders the task of archiving today's documents for prosperity all the more challenging.

The past decade has witnessed considerable activity in attempts at archival longevity (Laurie, 2000; Ross, 1999; Rothenberg, 1999, 2000, 2001; Waugh, 2001). Common approaches have been data migration, emulation and archiving of the original authoring software along with the archival documents themselves, all with limited success. Simply stated, long-term digital archiving involves the storage of unstable bit patterns on unstable media, for periods of time vastly exceeding the expected lifespan of that media, with subsequent (lossless) retrieval on obsolete equipment, (which may no longer be available) – quite a challenge!

Given the rapidly changing nature of digital records, can we devise a better approach to medium- to long-term archiving? It is the contention of the present study that the use of Open Source software can go part of the way to achieving this aim.

2. Digital Archiving Methods

The survival of a digital document relies on the ability to retrieve it from a digital archive and for it to appear as the author saw it at the time of creation. It encompasses not only physical, but also functional and content survival. Beagrie (2001) defines long-term preservation as encompassing two distinct but equally important functions: (i) long-term maintenance of a byte stream, and (ii) on-going access to its content through time and changing technology. Furthermore, the fundamental principle behind any digital preservation strategy must be to ‘do minimal harm’ (Waugh, 2000).

One approach to preservation is migration, defined as the periodic transfer of digital material from one hardware/software platform to another, or from one generation of computer technology to a subsequent generation (Muir, 2001). It has been used for decades to preserve operational data in data processing installations (Lorie, 2001). Another approach is emulation.

One difficulty with migration is that it requires a conscious act on the part of users. Time and financial constraints often cause the deferral of such tasks to a point in time when the old records deteriorate and become irretrievably lost (Rothenberg, 1999). The migration process proper involves physical changes to the stored data from the older format to the newer (involving binary representation, layout, encoding). The cumulative effect of repeated changes can however result in degradation of the evidentiary status and/or value

of the digital document (Thibodaux, 2001). Further, the migration (conversion) process itself can potentially degrade or corrupt the digital document (Rothenberg 2000; Waugh 2000; Henrard 2002).

With emulation, one (leading edge) computer system reproduces the behaviour of another (trailing edge) system. In this manner the emulator can run software designed for the (obsolete) system it is emulating (Rothenberg, 2000). Emulation can therefore be regarded as preservation based on technological advancement. Of necessity, it requires extensive knowledge of obsolete hardware architectures, operating environments and application software in order to build a virtual machine on a modern system environment.

Lorie (2001) developed a prototype in order to demonstrate the long-term viability of emulation for digital preservation, based on a "Universal Virtual Computer". This UVC approach stores the technical specifications for a simple, software-defined decoding machine in paper form, which potentially at least could last for centuries (silverfish, flood and fire notwithstanding). The key word here is 'simple' – it remains to be seen whether such detailed specifications can indeed be distilled into a brief paper document. Rothenberg (2000) developed an emulation system which stored the original (proprietary) authoring software along with the digital documents and a detailed human-readable description of the architectural environment of the current machine (in order to facilitate the development of emulators by *future* users). Other, more esoteric, approaches have been suggested, including buckets (Nelson, 2001), VERS (Waugh, 2000) and genomic storage (Wong, 2003).

3. Why Open Source?

Are migration, emulation or other techniques capable of delivering medium- to long-term archival storage? Let us consider each in turn.

Firstly, after several successive migrations (or refreshing and conversions), a document may bear little resemblance to the original (Muir, 2001). Worse, it may be impossible to subsequently determine just what has been lost (Waugh, 2000). Many organizations manage their information poorly, with the result that the migration process misses significant amounts of information (Waugh, 2000). Also, there is often a temptation to delay migration too long, past the point where information can be converted in a *cost effective* manner. Even if migration is timely and diligent, it can be less than effective because the target format is not 100% compatible with the source format (for example, a spurious comma in a comma-delimited spreadsheet could throw the entire document into disarray).

If migration is unsatisfactory, then perhaps adapting modern equipment to run old software (i.e. emulation) is. We face a new set of challenges with this approach. Invariably, software is Operating System (and often hardware) specific (Muir, 2001). Our options, therefore, are reduced to: (i) OS preservation (ii) hardware preservation (iii) OS simulation or (iv) machine simulation. Unfortunately the latter approach is often only a "best guess" effort, in the absence of museum machines for comparison purposes. Indeed, Bearman (1999) ponders whether preserving the OS and/or machine is preserving the wrong thing; in other words, is the target of preservation the system or the document?

The emulators of Lorie and Rothenberg both required storage of detailed system specifications along with the archival document(s). As systems become ever larger and more sophisticated, this becomes more and more of a challenge, and amounts to an exercise in complexity management. As Franz (1993) has observed though, developers need to overcome such limitations as:

1. Incompatible paradigms (e.g. object-oriented *versus* procedural),

2. Contrasting abstractions (e.g. Oberon differentiates between a data file and its access mechanism; Macintosh doesn't), and
3. The target system may have no equivalent constructs (e.g. a limit on the number of concurrently open files).

Buckets (Nelson, 2001) are storage units that contain data and metadata, as well as methods for accessing both. They are aggregative, intelligent, internet-accessible digital objects optimized for publication in Digital Libraries, and follow a Smart Object, Dumb Archive model. The tightly integrated nature of buckets severely limits their tolerance to change. For example, system changes require pointers to application and document locations within the bucket construct to be updated. Over time, this collection of pointers can become entangled and the risk of loss escalates. Failure to maintain the system can lead to data loss, context loss, reference loss and eventual loss of the entire document.

VERS objects encapsulate the information to be preserved within metadata that describes aspects of this information, such as data formats and descriptions of the preserved information (Waugh, 2000). In short, VERS is designed to be self-documenting and self contained. From the perspective of digital archiving though, VERS is limited because it constrains users to a specific set of (proprietary) application software, running on a (proprietary) OS.

Genomic storage (Wong, 2003) has created considerable interest but the need for a wet laboratory and the as yet unproven reliability of genomic storage (for example, random mutations in and non-survival of the host bacteria organism) renders it an unlikely candidate for long-term archival storage.

It is clear that all the above approaches to digital archiving are flawed. What then can be proposed as an alternative? Previous efforts have tended to seek a single, unique solution (i.e. panacea). As well, they tend to rely heavily on periodic user intervention. Not surprisingly, the goal has proved elusive. The approach taken in this study revolves around the use of Open Source software, which we believe has not previously been used in the context of medium- to long-term digital archiving. Further, we adopt the Future Digital Archiving (FDA) philosophy of facilitating the recovery effort where preservation either fails or is likely to fail.

Proprietary software is often expensive, requires periodic upgrading and is only available in executable, not (ASCII text) source code form. By contrast, Open Source software is free and the source code readily available. The advantages of Open Source can be summarized as follows (Open Source Initiative – OSI: http://www.opensource.org/licenses/ last accessed March 19 2004):

- the ability to view and edit source,
- the ability to port the application from one platform to another,
- the ability to modify the software functionality to suit specific user requirements, and
- the ability to store, copy and install applications at will without a licensing fee.

Under the General Public License (GPL), ported source code must also be published as Open Source; modified source code may or may not be published as Open Source, but may not be rendered proprietary. The flexibility afforded by the GPL licensing system, coupled with the availability of application source code, renders Open Source software uniquely suited to medium- to long-term archiving. The ability to edit/modify source gives it the longevity needed for the survival of digital documents beyond the current technology era.

The rationale behind storing source code along with the document is consistent with the FDA premise of being the 'contingency of last resort', namely where there is the potential for future users to modify the source in order to render the archived document(s) compatible with contemporary systems.

The aim of this study was to assess the viability of Open Source software as a robust, low-cost, low overhead and widely available method for medium- to long-term digital document preservation, capable of transcending system and software obsolescence, while delivering to future users the documents as their authors saw them.

4. Experimental Results

In order to validate our Open Source approach to medium- to long-term digital archiving, a series of experiments were undertaken. These involved: (a) fetching and installing the relevant Open Source programs, (b) archiving documents on various media, (c) restoring the archived files, and (d) assessing the passage of time on the archived documents. All media deteriorate over time. For example, magnetic media is susceptible to temperature, humidity, and magnetic fields; Optical media can suffer from both oxidation of the internal reflective material and from fungal infection.>

4.1 <Open Source Application Software>

The primary experimental platform was a 696MHz Dell PIII Latitude L400 laptop computer running (Open Source) Mandrake V9.0 Linux (public release 2002). The Open Source applications – all obtained from http://sourceforge.net – used in this study were:

- Open Office (desktop publishing)
- Xpdf (PDF reader)
- Xine (video/DVD player)
- The Gimp (graphical editor)
- Galleon (web browser)
- XMMS (CD player)

These representative Open Source tools were all compatible with their proprietary equivalents, to a greater or lesser extent. The application which exhibited most variation from its proprietary counterpart was Open Office. Compared with MicroSoft's Office-XP suite, we experienced mixed results: MS-Word, Excel and Powerpoint documents could all be opened, viewed and manipulated within Open Office, with most of their formatting and functions intact, but MS-Word macros proved incompatible. In the opposite direction, 'swx' files could be readily exported to MS Office (.doc) from Open Office. The online help within Open Office is somewhat incomplete, terse and lacks context, compared with the step-by-step online help provided in MS Office. Open Office – unlike MS Office – does not offer integrated email or a calendar accessory. Lastly, Open Office does not support collaborative editing tools.

4.2 Archiving of Documents together with their Authoring Software

The collection of Open Source application software described in Sect.4.1 was stored on a variety of media, these being:

- (fixed) magnetic disks (hard drives),

- (removable) magnetic disks (both floppy and Zip disks)
- (removable) optical disks (CD-R)

In the case of the fixed magnetic medium, a multi-threaded PVM (Parallel Virtual Machine) program – Archiver – was developed in order to distribute the digital archive to free disk space on the network of thirty-six SUN Ultra Workstations.

All thirty-six network nodes are housed within a single laboratory within the School of Information Technology & Computer Science at the University of Wollongong. They connect to a CISCO switch by way of 10/100 twisted pair Ethernet, thence by 1Gigabit fibre optic cable to the building switch which is connected to the main University server. Each SUN Ultra node (workstation) boasts 30GB of unused disk space, since account users are allocated storage by NFS on a remote RAID. Thus the total available (distributed) archival storage space available for this study was 1.08TB. From the perspective of this study, these thirty-six SUN workstations constituted a form of RAID (Redundant Array of Inexpensive Disks).

"Archiver" supported the management, transport and access of archived documents over the network, and was based on a design used in climate modelling and other fields where large data sets need to be accessed over networks (Allcock, 2001). Multi-threading led to improved performance when transporting large amounts over data over the network. Apart from storing the digital archives *per se*, fragmented, redundant, compressed copies were also stored on the network disks (generated using the `RAR` compression tool (http://www.win-rar.com/). While the `RAR` compression software is proprietary, the `RAR` decompressor is Open Source and thus falls within the scope of this study.

Complete archives were stored on both the (fixed) networked and (removable) optical disks (which took three 700MB CD-Rs). A subset was archived to a 100MB Zip disk, and an even smaller subset to a 1.4MB 3.5" floppy disk. These archives comprised the following:

- personal documents accumulated over a 6-year period,
- files obtained from the world wide web (mainly electronic libraries, where permission is granted for personal/educational use),
- files available under public license, and
- promotional material publicly available on the internet.

During the experiments, a (simple ASCII text) central catalogue of the archived files was maintained, along with the network node on which each file resided as well as the appropriate viewing tool. Whenever the archive was modified this central catalogue was updated and copied to *all* network nodes.

Archive retrieval was document-based, with appropriate (Open Source) viewing tools being used to access the desired documents. Apart from accessing the archived documents in their raw form, the `RAR` decompressor was also used to retrieve the redundant, compressed archives also stored on the network. This decompression tool provided error recovery/correction, which is an essential aspect of any archiving effort, in order to maintain the viability of the archive. Using multiple, redundant copies can also provide a recovery mechanism, since future users would be able to mix and match fragments from the multiple copies in order to reconstruct an error free archive.

The following four measures were developed in order to both evaluate the retrieved archive and to assess the effect of noise:

1. lossiness of conversion (where applicable),

2. ability to edit the data files (except for PDF),
3. ability to save back to the original format, and
4. functionality retention (e.g. embedded formulae, macros, hyperlinks etc.)

A Boolean measure – 'true' or 'false' – was gathered for each applicable criterion. For example, if a file could be successfully opened without generating an error message, then 'no-lossiness' was marked as 'true'. Results obtained at various levels of introduced noise are discussed in the following Section.

4.3 Simulating the Passage of Time via Introduced Noise

In this series of experiments, we deliberately introduced varying degrees of noise into the archived files in order to mimic deterioration of the storage medium over time. This was based on the premise that increasing damage eventually leads to a file becoming unusable.

A program was written which opened each file in the archive, read its contents, added a predetermined amount of noise, then wrote the result to a modified file. Four levels of noise were tried – 3%, 7%, 11% and 18%. The lower levels simulated distributed noise caused by adverse storage conditions; higher noise levels corresponded more to severe damage, such as that resulting from mechanical (physical) damage to the storage medium (for example, scratches on the surface of a CD are contiguous rather than uniform/random). The noise manifested as inverting the bits in every thirty-third character (3%), every 14th character (7%), and so on. In the experiments conducted during the course of this study, *any* noise distribution could have been synthesized; we actually opted for salt-and-pepper (speckle) noise, since this corresponds to data dropout in real-world systems (and leads to 'salt-and-pepper' like spikes).

Archives comprised the following file types:

- MS-Word (.doc)
- Documents with embedded Cyrillic characters
- Documents with embedded graphics (.wmf; .gif)
- Documents with embedded images (.bmp;.gif;.jpg)
- Documents with embedded hyperlinks
- Documents with embedded tables
- Text only documents
- MS-Excel spreadsheets incorporating csv formulas
- Portable Document Format files, generated by a variety of applications (.pdf)
- MPEG4 video files (.mpg)
- QuickTime movie files (.mov)
- JPEG image files (.jpg)
- Audio files (various formats – e.g. .wav; MP3; WMA; Ogg Vorbis)
- Program source code (C; C++; Perl)

In general, and not surprisingly, we found that the higher the proportion of introduced noise, the more adverse the effect on the stored archive.

Apart from source files, all document types displayed resilience (tolerance) to 3% added noise. All files were successfully opened using appropriate Open Source viewing tools. One file only out of 13 audio files failed to play back *All* source code files failed to compile (needless to say, even a *single* bit flipped in an executable file renders it unusable – no such binary files were archived during this study, only source code).

Increasing the level of artificially introduced noise to 7% resulted in 5 out of 456 `.doc` files exhibiting discernible errors. Likewise, 1 out of 20 spreadsheets, 1 in 20 PDF, and over half the JPEG images (12 out of 20) all contained errors. 3 audio files failed, and all source code files produced compile errors. All files opened successfully using the appropriate Open Source viewing tools however.

11% introduced noise produced errors in 56 .doc files, 5 spreadsheets, 7 PDF files and 15 JPEGs. 13 audio files failed (2 of 4 `.mov` trailers failed to run), and once again all source code files produced compile errors.

The highest noise level – 18% – yielded errors in 105 `.doc` files, and a further 20 files failed to open altogether. All 20 spreadsheets and 20 PDF files contained errors, with 17 spreadsheets and 8 PDFs failing to open. 7 JPEGs contained visible errors, with 13 failing to open ("this is not a jpeg file"), 4 `.mov` movie trailers failed, as did 13 audio files. As previously, all source code files caused compile errors.

These results reveal a correlation between the amount of introduced noise and the proportion of adversely affected files, except in the case of source code files, where all files were affected (which when one thinks about it is to be expected). At some point – between 10% and 20% – most files become unusable. Even at the highest noise levels though, some file types remained useable – specifically simple text files and MPEG videos (although as regards the latter, some noisy frames in a 25 frame per second film will be indiscernible to the human eye, whereas others will be). More complex file formats, such as machine readable source files (especially those containing embedded binary executable routines) and spreadsheet files (with multiple worksheets and embedded formulas) were much more susceptible to noise, and hence prone to failure.>

4.4 Aiding File Restoration by way of Decompression

<Another set of experiments was undertaken to verify that backup compressed archives could assist in the recovery process. Indeed they did – the entire archive was able to be successfully rebuilt from compressed volumes. We therefore advocate the use of compression/decompression in any practical archival system, not with the aim of saving space but for the increased error tolerance that results. A recovery record contains up to 524,288 recovery sectors, each recovery sector being able to recover 512 bytes of damaged data. Generally speaking, the size of recovery record may be approximated using the following Equation 1:

((archive size)/256 + number of recovery sectors)*512 bytes…….Equation 1.

RAR has documented tolerance of up to 10% errors, or one lost volume in 10 of a fragmented archive (http://www.win-rar.com/).

5. Conclusions

Based on our experimental results, it was concluded that file format complexity and specific file use significantly reduce noise tolerance. The file types which were most impervious to introduced noise were characterized by:

- Simple encoding (e.g. simple text documents), and/or
- High levels of compression (coupled with approximation in the decompression algorithms).

Source code files – being machine readable ASCII text files – are very susceptible to introduced noise, leading to compile time errors; they nevertheless remained *humanly* readable. With both ASCII text files and MPEGs, the viewer is quite tolerant of errors. A programmer skilled in the art of debugging could correct source code errors to the point where the file once again complies. Since MPEG decoding utilizes approximation in decompressing image data, this means that if one or more frames are compromised the human eye will not discern this (short of losing the file headers which contain the encoding and frame rate). Analysis of the movie frame-by-frame *would* reveal the compromised images, however.

This study demonstrated the fragility of digital documents one may typically require to archive. It also confirmed the viability of using Open Source tools to retrieve files created by proprietary software. The one requirement of this approach, however, is that the IT person responsible for archiving needs to be not only a capable programmer but also, *au fait* with Open Source methods – in other words, a proficient Systems Administrator (although one might expect that such a responsibility would usually fall within a Systems Administrator's job specification in any case?).

It is clear that the use of an Open Source approach to digital archiving merits further attention. Only through a longitudinal study over a sufficient time frame will the effectiveness or otherwise of any approach to digital archiving be demonstrated. While beyond the scope of the present study, it is proposed that such a study be undertaken using an Open Source approach.

References

Allcock, B. et al. (2001) "High-Performance Remote Access to Climate Simulation Data: a Challenge Problem for Data Grid Technologies" *Proceedings of the ACM/IEEE Supercomputing Conference – SC2001*, Denver, CO, November, 2001.

Beagrie, N. et al. (2001) "Attributes of a Trusted Digital Repository" Draft Report OCLC/RLG, 31 August, 2001, pp.1-30.

Bearman, B. (1999) "Reality and Chimeras in the Preservation of Electronic Records" *D-Lib Magazine*, **5**(4).

Franz, M. (1993) "Emulating an Operating System on top of Another" *Software Practice & Experience*, **23**(6): 672-692.

Henrard, J. et al. (2002) "Strategies for Data Reengineering" *Proceedings of the Working Conference on Reverse Engineering – WCRE'02*, 29 October – 1 November, 2002, Richmond, VA. Piscataway, NJ: IEEE Computer Society Press, pp.211-220.

Kahle, B. (1997) "Archiving the Internet" *Scientific American*, March, 1997.

Lorie, R. (2001) "Long Term Preservation of Digital Information" *Proceedings of the 1st Joint ACM/IEEE Conference on Digital Libraries – JCDL'01*, June 24-28, 2001, Roanoke, VA, pp.346-352.

Muir, A. (2001) "Legal Deposit of Digital Publications: a Review of Research and Development Activity" *Proceedings of the 1st Joint ACM/IEEE Conference on Digital Libraries – JCDL'01*, June 24-28, 2001, Roanoke, VA, pp.165-173..

Nelson, M. (2001) “Buckets: a Digital Technology for Preserving NASA Research” *J. Government Information*, **28** (2001): 369-394.

Open Channel Software (2004) NASA COSMIC Collection, http://www.openchannelfoundation.org/cosmic

Ross, S. and Gow, A. (1999) “Digital Archaeology: Rescuing Neglected and Damaged Data Resources: A Study Within the Electronic Libraries (eLib) Programme on the Preservation of Electronic Materials” *Library Information Technology Centre, South Bank University*, February, 1999, pp1-83.

Rothenberg, J. (1995) "Ensuring the Longevity of Digital Documents" *Scientific American*, **272**(1) (1995): 42-47.

Rothenberg, J. (1999) "An Experiment in Using Emulation to Preserve Digital Publications" *The Koninklijke Bibliotheek & RAND-Europe*, April, 1999.

Thibodeau, K. et al. (2001) “Preservation Task Force Report” *International Research on Permanent Records in Electronic Systems (InterPARES)*: 3-14.

Waugh, A., Wilkinson, R., Hills, B. and Dell'oro, J. (2000) "Preserving Digital Information Forever" *Proceedings of the 5th ACM Conference on Digital Libraries*, June, 2000, San Antonio, TX, pp.175-182.

Wong, P.-C., Wong, K.-K. and Foote, H. (2003) “Organic Data Memory Using the DNA Approach” *Communications ACM*, **46**(1) (2003): 95-98.