Challenges to Identify Software for Reproduction of Complex Environments

Functional Longterm Archiving Group
Albert-Ludwigs University Freiburg
Hermann-Herder Str. 10
79104 Freiburg i. B., Germany

ABSTRACT

Digital research data and digital artefacts of various types do not just exist and run on their own, but depend on a certain runtime environment to make sense or render correctly. Reproducible science as well as reliable research or business workflows and processes depend on an unambiguous description of such environments. Ideally such software environments can be reproduced automatically to be used for emulation services. The requirements of different users of software differ and as the type of software is very diverse in the different domains. The challenge explicitly does not focus on description of environments and definition of metadata for them but on organizational aspects of how software should be managed in the long run.

1. INTRODUCTION

The information age has changed the economic landscape and research environments thoroughly. Nearly every piece of valuable scientific or business information is created, transferred and stored digitally. Moreover, the information is not necessarily bound to single, easily identifiable digital objects but can be embedded in complex (business) processes. Reconstructing and re-enacting these processes and associated digital artefacts, e.g. due to regulation or legislative requirements, a legal dispute or an audit, may be required at some point. Then, both reliability of reproduction and associated costs will matter.

Research data, business processes and generic digital artefacts could not be viewed or handled simply by themselves, but they require a specific software and hardware environment to be accessed or executed properly. Especially artefacts of many complex and domain specific formats are best handled by the matching applications they were created with. Independent of choosing a migration or emulation approach for long-term access, just storing those artefacts themselves will not suffice. Thus, software is required for a wide range of reasons even after its official end-of-live announced by the manufacturer. For this and other reasons, many institutions and companies need to preserve copies of software packages they once had in use.

In this paper we present a couple of challenges when faced with archiving complex digital environments and associated software. There exist a couple of organizational and technical issues with software archiving. Different types of software and installations are required for different purposes. This leads to a couple of research and organizational questions:

- How to properly describe and distinguish software (packages) like for localized variants and different versions?
- Which (institutional, commercial, private) users and services may profit from a software archive?
- How should software be preserved (centralized vs. decentralized approach)?
- How to prove the authentic reproduction of the requested environment?
- What should standardized workflows look like for software preservation?
- What should a generic software archive/museum service provide (installation media, (preservation) metadata on software, installation packages, licenses)?

We see two approaches to the problem: The technical issues as addressed in PREMIS, KEEP and/or TOTEM[1] and the institutional perspective: Some relevant institution(s) should keep the standard software components and these should be uniquely referenced, like file identifiers in PRONOM.

For at least a significant proportion of the software to be covered, licensing might complicate the whole issue as organizations and entities have different software licensing contracts with different companies.

2. TYPES OF SOFTWARE

Institutions and users have to decide which software needs to preserved how and by whom. It depends on the intended use cases. In simpler cases just some standard office or business environments with standard software are needed to render preserved artefacts in emulated original environments. Complex cases could be composed of very special non-standard, custom-made software components from non-standard sources like for development systems or from complex business processes.

Software components required to reproduce original environments for certain (complex) digital objects can be classified in several ways. There is the standard software like operating systems and off-the-shelf applications sold in (significant) numbers to customers. There might exist different releases and various localized versions (the user interaction part translated to different languages like for Microsoft Windows or Adobe products) but otherwise the copies were exactly the same. In general it does not really matter if it

[1]How to describe (software and hardware) environments, how to (re)create them etc.

is a Dutch, English, or German Word Perfect to render a document. But for the user dealing with it or an automated process like used for migration-through-emulation [6] the different labeling of menu entries and error messages matters.

The concept of explicit is somewhat different for Open Source or Shareware-like software. Often there are much more "releases" available as the software usually gets updated permanently and does not necessarily have a distinct release cycle. Different, to much deprecated commercial software the open source packages feature full localization, as they did not need to distinguish different markets.

In many domains custom made software and user programming plays a significant role. This could be scripts or applications written by scientists to run their analysis on gathered data, run specific computations, or extend existing standard software packages. Other examples are software tools written for governmental offices or companies to produce certain forms or implement and configure business processes. Such software is to be taken care of and stored alongside the preserved object. The same applies for complex setups of standard components with lots of very specific configurations.

If such standard software is required, it would make sense to identify it uniquely. This would help to de-duplicate efforts to store copies. Even if a memory institution, commercial service to reproduce original environments maintains its own copy, it does not necessarily need to replicate if other copies are already available somewhere. Additionally, it simplifies to reproduce environments in an efficient way.

Not for all software components a (federated) software archive of standard components makes sense.

3. WHAT SHOULD BE IDENTIFIED?

There are a couple of ideas on software identification and description already discussed for the upcoming PREMIS 3.0 standard regarding *environments* [3]. Suitable persistent identifiers would be definitely helpful to tag software like ISBNs or ISSNs describe books and other media. These tags would be useful for tool registries like TOTEM [4] as well or should match to PREMIS PUIDs [2]. There could be three layers of IDing become relevant:

- On the most abstract layer a software is described as a complete package, e.g. Windows 3.11 US Edition, Adobe Page Maker Version X or Command & Conquer II containing all the relevant installation media, license keys etc. The ID of such a package could be the official product code or derived from it. Nevertheless it might be difficult to distinguish between hidden updates. During the software archiving experiment at Archives New Zealand we got two different package sets of Word Perfect 6.0.

- On the layer of the different media (relevant only if it is not just one downloaded installation package) each floppy disk or each optical medium could be distinguished. E.g. Windows 3.11 as well as applications like Word Perfect provided different disks with just printer drivers on it. The CD 1 or 2 in Command & Conquer differentiated which adversary in the game you were assigned to.

- On the individual file layer executables, libraries, helper files like font-sets etc. could be distinguished. The number of items on this set is the largest. The approach to run a collection of digital signatures of known, traceable software applications is followed e.g. by the NSRL.[2]

Usually it is not trivial to map the installed files to files on the installation medium, as the files get typically packed on the medium and a couple of files get created during the installation procedure.

Depending on the actual goal, the focus of the IDs is different. To actually derive what kind of application or operating system is installed on a machine, the file level is relevant. To just reproduce a certain original environment (for e.g. emulation) the package level is more interesting. In certain cases it might be interesting to address a single carrier, e.g. to automate installation processes of standard environments consisting of an operating system plus a couple of applications.

For the description of software and environments should be checked what could be learned from commercial software installation handling and lifecycle management. Large institutions and companies have well-defined workflows to create software environments for certain purposes.

4. SOFTWARE ARCHIVE

What should be archived and who are the stakeholders and users. How can the archive be supported?

A model for nearly full-archiving of a domain is the Computer Games Museum in Berlin which receives every piece of computer game which requires an USK[3] classification. The collection is supplemented by donations of a wide range of software (operating systems, popular non-gaming applications) and hardware items (computers, gaming consoles, controllers). Thus, the museum acquired a nearly complete collection of the domain. An upcoming problem is the rising number of browser and online games which never get a representation of a concrete medium. Another unresolved issue is the maintenance of the collection. At the moment the museum does not have enough funds for bitstream preservation and proper cataloguing the collection.

Archiving (of standard software) already takes place, for example, at the Computer History Museum, the Australian National Library, the National Archives of New Zealand or the Internet Archive to mention a few. Unfortunately, the activities are not coordinated. Both the mostly "dark archives of memory institutions" and the online sites for deprecated software of questionable origin are not sufficient for a sustainable strategy. Nevertheless, landmark institutions like national libraries and archives could be a good place to archive software in a general way. Nevertheless, the archived software is of any use only, if properly described with standard metadata. Ideally, the software repositories provide APIs to communicate with the software archive and attach services to it. The service levels could differ from just offering metadata information to complete software packages. As an addition to the basic services museums could offer interactive access to selected original environments, as there

[2]National Software Reference Library, see `http://www.nsrl.nist.gov/`

[3]USK is the German abbreviation for the Entertainment Software Self-Regulation Body, an organisation which has been voluntarily established by the computer games industry to classify computer games, see `http://www.usk.de/en/`.

is a significant difference between having a software package just bit-stream preserved and have it available to explore and test it for a particular purpose interactively. Often, specific, implicit knowledge is required to get some software item up and running. Archiving institutions like museums could try to build online communities around platforms and software packages. Life "exhibition" of software helps community exchange and can attract users with knowledge who would be otherwise difficult to find.

Software museums can help to reduce duplicated effort to archive and describe standard software. It can at least help that not every archive needs to store multiple copies of standard software but simply can refer to other repositories. Software museums or archives could become brokers for (obsolete) software licenses. They could serve as a place to donate software (from public, private entities), firmware and platform documentation. Such institutions could simplify the proceedings for a software company to take care of their digital legacy. A one-stop institution might be much more attractive than to negotiate license terms of legacy packages with multiple stakeholders.[4] Software escrow services [5] can complement the activities. A museum can operate in different modes like in a non-for-profit branch for public presentation, community building, education etc. and commercial branch to lend/lease out software to actually reproduce environments in emulators for commercial customers.

The situation could be totally different for research institutions and users of custom made software. Such packages do not necessarily make sense in a (public) repository. In such cases the question of, how the licensing will be handled arises. If obsolete, they could be handed over to the archive managing the research primary data.

Another issue is the handling of software versions. Products are updated until announced end-of-live. Would it be necessary to keep every intermediate version or concentrate on general milestones. An operating system like "Windows XP" (32 bit) was officially available in several flavors (like "Home" or "Professional") from 2001 till 2014. In many cases a "fuzzy matching" would be acceptable as a certain software package runs properly in all versions. Other software might require a very specific version to function properly. This needs to be addressable (and could be matched to the appropriate PRONOM environment identifiers). Plus, there are a couple of preservation challenges in the software lifecylce [1].

5. DISCUSSION

The following questions could be discussed:

- Does it make sense (at all) to run a centralized software archive in a relevant size, assuming that for modern, complex scientific environments, the software components are much too individual? What kind of software would be useful in such an archive? Which versions should be kept?
- Would it be possible to establish a PRONOM-like identifier system (agreed upon and shared among the relevant memory institutions)?
- How, through which APIs should software and/or metadata be offered (or ingested)? How should the software archive adapt to the ever changing form of installation media from tapes, floppies to optical media of different types to solely network based installations? Would it be possible to run the software archive as a backend, where locally ingested software is stored in the end?
- Is the advantage gain of centralizing knowledge and storage of standard software components big enough to outweigh the efforts required to run such an archive?
- Do proper software license and handling models exist for such an archive, like donation of licenses, taking over abandoned packages, escrow services? Would it be possible to bridge the diverse interests of diverse users of a diverse range of software and software manufacturers?
- On which level should a software archive be run: Institutional (e.g. for larger (national) research institutions, state or federal or global level?

[4]Software companies should have a positive attitude towards such a platform or lawmakers should push it a bit.

6. REFERENCES

[1] J. Barateiro, D. Draws, M. A. Neuman, and S. Strodl. Digital preservation challenges on software life cycle. In *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pages 487–490. IEEE, 2012.

[2] P. Caplan. Understanding premis. *D-Lib Magazine*, 15(3/4), 2009.

[3] A. Dappert, S. Peyrard, J. Delve, and C. C. Chou. Describing digital object environments in premis. In R. Moore, K. Ashley, and S. Ross, editors, *Proceedings of the 9th International Conference on Preservation of Digital Objects (iPRES2012)*, pages 69–76, 2012.

[4] J. Delve and D. Anderson. *The Trusted Online Technical Environment Metadata Database – TOTEM.* Verlag Dr. Kovac, 2012.

[5] K. Hobel and S. Strodl. Software escrow agreements. In *International Legal Informatics Symposium (IRIS)*, 2 2012.

[6] D. von Suchodoletz, K. Rechert, R. Welte, M. van den Dobbelsteen, B. Roberts, J. van der Hoeven, and J. Schroder. Automation of flexible migration workflows. *International Journal of Digital Curation*, 2(2), 2010.