Future-Proof Preservation of Complex Software Environments

Klaus Rechert, Dirk von Suchodoletz and Isgandar Valizada
Department of Computer Science
University of Freiburg, Germany

ABSTRACT

Emulation evolves into a mature digital preservation strategy providing authentic access to a wide range of digital objects using their original creation environments. In contrast to migration, an emulation approach requires a number of additional components, namely the full software-stack required to render a digital object, and its configuration. Thus, independent of which emulator is chosen, contextual information of the original computer environment is always needed.

To overcome this knowledge gap, a formalization process is required to identify the actual building blocks for an authentic rendering environment of a given object. While the information gathering workflow relies heavily on user knowledge and manual interaction during ingest, the workflow is coupled with a feedback loop so that both a complete emulation environment and preservation of desired properties for later access are ensured.

1. INTRODUCTION

In most cases the best way to render a certain digital object is using its creating applications, since these cover most of the object's significant properties and hence, provide an authentic and possibly an interactive user experience. Therefore, emulation is a key strategy to provide a digital object's native environment and thus to maintain its original "look" and "feel" [7]. In some cases the existence of access alternatives, e.g. format migration, is not guaranteed due to the proprietary nature of the object's file formats, or even impossible due to the complex structure of the object (e.g. digital art, computer games, etc.).

Recreating software environments using emulation requires detailed knowledge about the objects' dependencies (e.g. operating systems, libraries, applications). Viewpaths (VP) [6] represent an ordered list of such dependencies for a given object, defining an order for the sequence in which these dependencies are required. Hence, in combination with a comprehensive and well-managed software archive any ancient computer environment could be rebuilt. Nevertheless this topic is still largely neglected by practitioners and research communities in the digital preservation domain.

If a digital object becomes subject to digital preservation, a defined workflow is required to support the preservation process of the object's original context i.e. rendering environment. The workflow makes use of the user's knowledge to identify necessary components of the object's rendering environment, to the effect that the rendering environment is complete and there are no dependency conflicts, at least for the chosen configuration and the digital object's contextual environment. For current computer environments and applications plenty of user knowledge is available. Thus, the project's proposed workflows focus on users "owning" a system setup e.g. for performing business of scientific processes. More specifically, owners of today's digital objects have good knowledge of the object's properties, their desired functions and utility, at least to extent of the objects' original purpose. Furthermore, preserving the knowledge of installation, configuration, and usage of software components ensures the recreation process of past system environments. By providing a preview of the emulated and recreated environment during ingest the user is able to test if the chosen setup meets the desired rendering quality and functionality. Figure 1 shows the proposed workflow in an abstract way.

2. RELATED WORK

In order to preserve a digital object's rendering environment, any dependencies from interactive applications to operating system and hardware components need to be identified. A widely adopted method which is integrated as a service in many digital repositories and institutional archives is the file type database PRONOM [2]. But identifying files and linking applications to them is only the first step. Several tools were proposed to resolve software dependencies from platform specific object-code binaries. E.g. DROID [1] makes use of "file-magic" fingerprints in combination with a database, others make use of system library resolving mechanisms [4]. While these tools and techniques provide useful information and hints to the users, they do not guarantee the generation of a suitable rendering environment, for instance, regarding completeness, quality and conflicting dependencies. In case of database dependent tools, appropriate data for a specific digital object is required.

A significant challenge when dealing with outdated software packages is the diminishing knowledge of how to han-

[1]DROID Project, `http://droid.sourceforge.net/`, (5/28/2012).

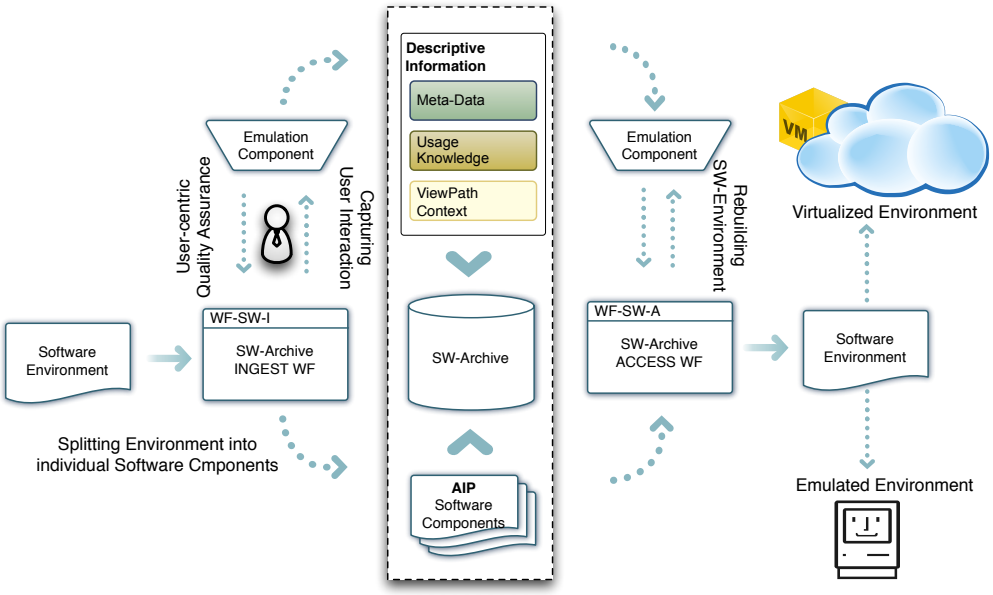

Figure 1: Preservation of complex software environments

dle the installation and configuration processes properly. One method to leverage the effort and archive the required knowledge is to automate the different installation steps for each relevant package. A viable approach is illustrated by Woods and Brown, who describe a software designed to minimize dependency on this knowledge by offering automated configuration and execution within virtualized environments [10]. This group demonstrated how to deploy automation scripts, i.e. GUI automation in order to install applications on demand. Their approach was successfully tested on different applications in several Windows versions. However, the script language used requires programming skills and in-depth knowledge of the operating system. Within earlier work of the same focus Reichherzer and Brown addressed the creation of emulator images suitable to render Microsoft Office Documents [5].

3. PRESERVATION OF COMPLEX SOFTWARE ENVIRONMENTS

In contrast to a migration strategy, the emulation approach requires a number of additional components and configurations to provide access to digital objects. Thus, independent of which type of emulator is chosen, contextual information of the computer environment is always required. To overcome this gap of missing knowledge, a formalization process is required to compute the actual building blocks for an authentic rendering environment of the digital object.

The VP model describes a system environment starting from the rendering application of the digital object to the description of required software and hardware requirements. If one of these requirements is not met (e.g. hardware components are not available), emulators can be used to bridge the gap between the digital past and future contexts. VPs define an abstract model which can be instantiated by an applicable workflow. To complete the process and compute dependencies technical metadata on various layers is required [3]. Descriptive information needs to be extended, for instance by adding information on required applications, suitable operating systems, and emulators. Additionally, software archiving is required to be able to reproduce complete original environments. Software archives play a vital part in an emulation-centric preservation approach as deprecated software products, legacy hardware drivers, older font-sets, codecs and handbooks for the various programs will become more and more difficult to find.

3.1 Ingest Workflow

Software components need to be preserved and enriched with additional information (meta-data), like operation manuals, license keys, setup how-to's, and usage knowledge. Furthermore, each software component defines its own soft- and hardware dependencies. To ensure long-term access to digital objects through emulation, not only the availability of technical meta-data (e.g. TOTEM entities [1]) are required, but these VPs also need to be tested and evaluated by users aware of the digital object's environment properties and performance. Hence, a defined workflow is required which allows the user to (pre-)view and evaluate the rendering result of each step of VP creation. This can be achieved by providing a framework to perform a structured installation process of a reference workstation. Figure 2 presents a functional flow diagram of the suggested workflow for the addition of new software components to the software archive.

1. The ingest workflow starts with the import of a single software component (WF-I-SW-0). This component might be available through a reference to the digital object already contained in a AIP/DIP container of some digital archive. Otherwise the user is able to upload the object from the local filesystem.

2. In a second step (WF-I-SW.1) the user is able to provide a detailed descriptive information of the object. This description is used as archival meta-data for indexing and search purposes.

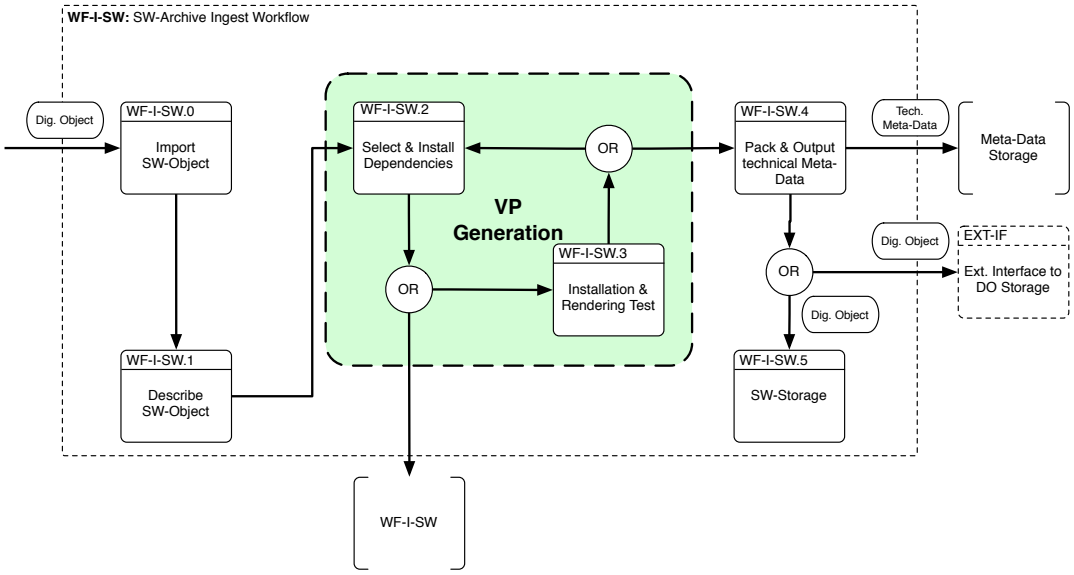

Figure 2: Ingest workflow of a single software package.

3. At workflow step WF-I-SW.2 the user is able to select the software component's hard- and software dependencies. The possible choices are assembled based on already existing knowledge of the software archive or by using external sources.If all required dependencies of the object already exist, the user is able to proceed to workflow step WF-I-SW.3. If the required dependency is not known or not available in the software archive, it must first be ingested into the software archive by using a recursive invocation of the ingest workflow for this missing dependency software component.

4. The options of workflow step WF-I-SW.3 depend on the type of the software component. If it is an operating system, it is run by means of emulation and the user is able to interact with the corresponding environment. If the type of the software component is installable software suitable for a certain operating system (e.g. library, driver, application), it is injected into the system and its installation (either by manual interaction or in an unattended manner) is performed. After the installation is finished, the user is able either to confirm a successful installation or to reject it in case of failure. A successful installation implies automatic extension of the VP for this software component with a new dependency object. Thus after each dependency object is confirmed to have been successfully installed, the VP is extended accordingly until no more dependencies are required for this software component. The resulting VP then represents a suitable manually tested and confirmed rendering environment. If the installation fails due to missing software or hardware dependencies the user has to change the VP accordingly. A repetition of tasks at step WF-I-SW.2 may be required for this.

5. If the user reached step WF-I-SW.4 of the workflow, a suitable VP has been built and a technical meta-data (VP) has been generated. The generated meta-data information might consist not only of the VP but also of user feedback about the quality and/or costs of the produced technical metadata.

6. In a final step the software component is submitted for further processing as SIP to a software archive.

The proposed workflow requires significant manual user interaction and seems costly and time consuming at first sight. However, regarding preservation of current digital objects, the basic rendering environment is quite stable concerning software and hardware dependencies. Usually the main differences can be found on the top layer of the VP description, i.e. only a few additional steps are required if the software archive already contains suitable VP descriptions of today's common digital objects. The ingest workflow could be further accelerated by employing caching strategies on created software images and by automation of installation tasks.

In order to automate such processes, unattended user interactions with an operating system have led to an interesting possibility of performing automatic dependency installations. So called interactive session recorders are able to record user interactions such as mouse clicks/movements and keystrokes performed by the user during the interaction with an operating system and save them to an interactive workflow description (IWD) file. The interactive session replayers on the other hand are able to read the IWD files and reproduce these actions. Applying this technique to the ingest workflow of the software archive implies recording of all input actions performed by the user during the installation of a software component and saving this information for future purposes. The attractiveness of this approach is that no additional programming must be done in order to automate the installation process, which makes this approach available to a wide range of users and computer systems. Furthermore, the IWD approach is independent of the GUI system used and the underlying operating system [9]. Thus for any successful run of the proposed ingest workflow meta-

data (VP) is generated as

$$\begin{aligned} VP_0 &= < emulator, OS > \\ &\dots \\ VP_n &= < VP_{n-1}, IWD_n, SW_n > \end{aligned}$$

starting with an emulator / operating system combination which is successively extended by a software component (referenced as TOTEM entity) and the associated installation and configuration routine.

The combination of base images made for a certain emulator plus a software archive of all required VP software components enriched with knowledge of how to produce a certain original environment (on demand) provides the necessary base layer for the future access of original artifacts. The additional costs in terms of manual interaction during object ingest are able to reduce the long-term preservation planning costs, since only the bottom layer (i.e. emulator) of the VP needs to be taken into account.

3.2 Access Workflow – Rendering of Software Environments

Having a complete VP description for an object is certainly not sufficient for it to be accessed, i.e. rendered. A suitable environment is to be recreated first. In this paper we refer to the process of recreating such an environment by using the term *viewpath instantiation*. A VP is considered as instantiated if the operating system contained in the VP description is started, successfully booted and available for external interaction through the emulated input/output devices. Furthermore, all remaining dependencies defined in the VP for the object need to be installed.

The proposed workflow delegates the task of VP instantiation to a separate abstract service: the emulation component. In order to allow a large, non-technical user-group to interact with emulators an abstract emulation component has been developed to standardize usage and hide individual system complexity. Each Web service endpoint provides a ready-made emulator instance with a remote accessible user interface (currently VNC and HTML5 web output are supported). Furthermore, standard system interaction is available, such as attaching/detaching removable drives (e.g. floppies, CD/DVDs) and attaching hard-drives and images to an emulator. A full description of a distributed emulation setup was presented in earlier work [8].

4. CONCLUSION & OUTLOOK

Emulation becomes a more and more accepted and mature digital preservation strategy to provide access to a wide range of different objects. As it does not demand any modification of the objects over time, the objects do not need to be touched unless requested.

The proposed approach defines a user-centric workflow, which makes use of current user knowledge and thus is able to provide certain guarantees regarding completeness, rendering quality, and non-conflicting dependencies. Furthermore, through a defined framework all interactions between user and computer environment could be observed and recorded. Thereby, not only a more efficient VP instantiation is possible but also knowledge on the usage of certain computer environments and their software components can be preserved. While an emulation approach has technical limitations (e.g. due to external (network) dependencies, DRM, license dongles, etc.), the proposed workflow is able to uncover such issues and indicates risks w.r.t. to long-term preservation.

With the development of a defined work process and associated workflows the groundwork for system integration and automation has been made. With more user experience and feedback, workflow-components suitable for automation could be identified, designed and implemented.

Acknowledgments

The work presented in this publication is a part of the *bwFLA – Functional Long-Term Access*[2] project sponsored by the federal state of Baden-Württemberg, Germany.

5. REFERENCES

[1] D. Anderson, J. Delve, and D. Pinchbeck. Towards a workable, emulation-based preservation strategy: rationale and technical metadata. *New review of information networking*, (15):110–131, 2010.

[2] T. Brody, L. Carr, J. M. Hey, and A. Brown. Pronom-roar: Adding format profiles to a repository registry to inform preservation services. *International Journal of Digital Curation*, 2(2), 2007.

[3] R. Guenther and R. Wolfe. Integrating metadata standards to support long-term preservation of digital assets: Developing best practices for expressing preservation metadata in a container format. In *Proceedings of the 6th International Conference on Preservation of Digital Objects (iPRES2009)*, pages 83–89, 2009.

[4] A. N. Jackson. Using automated dependency analysis to generate representation information. In *Proceedings of the 8th International Conference on Preservation of Digital Objects (iPRES2011)*, pages 89–92, 2011.

[5] T. Reichherzer and G. Brown. Quantifying software requirements for supporting archived office documents using emulation. In *Digital Libraries, 2006. JCDL '06. Proceedings of the 6th ACM/IEEE-CS Joint Conference on*, pages 86–94, june 2006.

[6] J. van der Hoeven and D. von Suchodoletz. Emulation: From digital artefact to remotely rendered environments. *International Journal of Digital Curation*, 4(3), 2009.

[7] R. Verdegem and J. van der Hoeven. Emulation: To be or not to be. In *IS&T Conference on Archiving 2006, Ottawa, Canada, May 23-26*, pages 55–60, 2006.

[8] D. von Suchodoletz, K. Rechert, and I. Valizada. Remote emulation for migration services in a distributed preservation framework. In *Proceedings of the 8th International Conference on Preservation of Digital Objects (iPRES2011)*, pages 158–166, 2011.

[9] 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.

[10] K. Woods and G. Brown. Assisted emulation for legacy executables. *International Journal of Digital Curation*, 5(1), 2010.

[2] bwFLA – Functional Long-Term Access, `http://bw-fla.uni-freiburg.de`.