The International Journal of Digital Curation
Issue 1, Volume 6 | 2011

Automation of Flexible Migration Workflows

Dirk von Suchodoletz, Klaus Rechert, Randolph Welte,
University of Freiburg

Maurice van den Dobbelsteen, Bill Roberts
National Archives of the Netherlands

Jeffrey van der Hoeven,
Koninklijke Bibliotheek

Jasper Schroder,
IBM Netherlands B.V.

Abstract

Many digital preservation scenarios are based on the migration strategy, which itself is heavily tool-dependent. For popular, well-defined and often open file formats – e.g., digital images, such as PNG, GIF, JPEG – a wide range of tools exist. Migration workflows become more difficult with proprietary formats, as used by the several text processing applications becoming available in the last two decades. If a certain file format can not be rendered with actual software, emulation of the original environment remains a valid option. For instance, with the original Lotus AmiPro or Word Perfect, it is not a problem to save an object of this type in ASCII text or Rich Text Format. In specific environments, it is even possible to send the file to a virtual printer, thereby producing a PDF as a migration output. Such manual migration tasks typically involve human interaction, which may be feasible for a small number of objects, but not for larger batches of files.

We propose a novel approach using a software-operated VNC abstraction layer in order to replace humans with machine interaction. Emulators or virtualization tools equipped with a VNC interface are very well suited for this approach. But screen, keyboard and mouse interaction is just part of the setup. Furthermore, digital objects need to be transferred into the original environment in order to be extracted after processing. Nevertheless, the complexity of the new generation of migration services is quickly rising; a preservation workflow is now comprised not only of the migration tool itself, but of a complete software and virtual hardware stack with recorded workflows linked to every supported migration scenario. Thus the requirements of OAIS management must include proper software archiving, emulator selection, system image and recording handling. The concept of view-paths could help either to automatically determine the proper pre-configured virtual environment or to set up system images for certain migration workflows. View-paths may rise in demand, as the generation of PDF output files from Word Perfect input could be cached as pre-fabricated emulator system images. The current groundwork provides several possible optimizations, such as using the automation features of the original environments.[1]

---

[1] This paper is based on the paper given by the authors at the 6th International Conference on Preservation of Digital Objects (iPres 2009), October 2009; received January 2010, published March 2011.

Introduction

The creation of most digital objects occurs with interactive graphical user interfaces, available at that particular time period. For example, an important long-lasting spare part for the aviation industry may have been designed with a specific CAD program, which has now disappeared from the market; or a governmental organization may deposit a large amount of text, spreadsheet documents and images (Figure 1). Last but not least, a museum may inherit the legacy of a famous author, who produced a large number of manuscript files with a word processor on the Amiga, a now obsolete personal computing platform. Archiving and preservation organizations already have a large quantity of such objects in various types and the quantity is growing. The organisations not only have to make the objects available to their users, they have to safeguard the digital longevity of them. The long-lasting preservation of digital objects poses completely different requirements from those of the original creation of the objects. Consequently, at some point the organisations will need to make a presentation copy of the objects and convert the objects to a current, sustainable file format. Due to the scale, the only financially and organisationally feasible way of supporting both actions, thereby achieving both goals, is automated processes and workflows at different levels. We show what kind of work has to be done to tackle this problem.

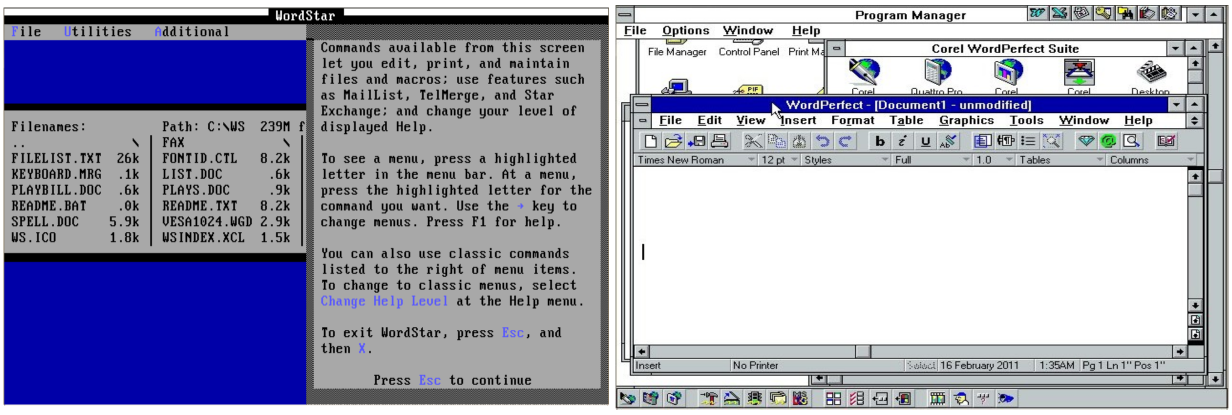
Given the challenges previously outlined, we are suggesting a new method, which uses an operating system and application-independent interaction workflow for the migration of digital objects using interactive software in an emulated or virtualized environment. The abstraction of machine start and shutdown, as well as the interaction with the running software, allows us to automate tasks originally done by humans. The first part of the paper discusses primary challenges and presents some first experimental results. The second half discusses additional requirements for the OAIS perspective of archive management. The proposed automation approach assumes the existence of prepared virtual or emulated machine images with the required software stack for a certain object migration workflow. However, especially for past digital objects, those environments do not necessarily need to exist any more. They have to be reproduced from the original software using an appropriate emulated hardware. Here the concept of view-path, together with certain significant properties of the object, could help define the software requirements.

The Migration Tool Challenge

A simplified digital preservation migration service is an engine receiving one or more digital objects as input and producing one or more digital objects as output. The information on the input and output formats is provided together with the input object by the calling procedure (Rechert et al., 2010). This procedure is rather straight forward for command line tools, like ImageMagick, and used, for instance, as a converter for BMP into PNG images. Such a class of tools exists only for very popular, open and well-described formats, but unfortunately not for all formats in the digital collections today's memory institutions have to cope with. Thus, a substantial problem in creating automated migration processes is the availability of suitable tools.

The approach we suggest takes another angle compared to the traditional command-line-based batch processing. It makes the assumption that a digital object is, in most cases, best rendered in its original environment. Emulation is able to provide near to the original or compatible environments for such deprecated hardware and software stacks (von Suchodoletz & van der Hoeven, 2009). However, many of the applications in use at that time were designed as interactive software, most of them without interfaces for automation. This means that graphic, product design, audio/video or word processing programs cannot perform basic tasks, such as the opening and saving of a file in another format, as an unattended and fully automated task. If larger amounts of objects are to be processed, it isn't efficient to employ humans for those migrations. Moreover, human interaction in such a repetitive task is error-prone and expensive. The attempt to add new functions to an application is generally very complex and sometimes even impossible, since the source-code and the required knowledge are no longer available (van der Hoeven, van Diessen & van der Meer, 2005). For a large number of different applications the cost would be very high and would only be worth the effort for very popular formats. Also, it is becoming increasingly difficult to find suitable staff familiar with the deprecated environments of older computer systems. Many current applications offer a certain range of file format importers, but it is not possible to rely solely on their availability. For instance, no modern text processor is able to load Word Perfect or AmiPro documents reliably.

The traditional approach to help the user automate interactive tasks to a certain degree is through a so-called macro-recorder. These are specialized tools or functions of an application or operating system user interface that capture sequences of actions carried out, e.g. the creation of a new file, opening the address database and selecting an entry, copying text, saving or printing the file, or the creation multiple similar letters. However, the automation features are not standardized in terms of usability and features (sets). Special tools might be needed, as well as a deeper knowledge of the applications and operating systems.



```
Filenames:          Path: C:\WS   239M f
..              \   FAX              \
FILELIST.TXT  26k   FONTID.CTL     8.2k
KEYBOARD.MRG   .1k  LIST.DOC        .6k
PLAYBILL.DOC   .6k  PLAYS.DOC       .9k
README.BAT     .0k  README.TXT     8.2k
SPELL.DOC     5.9k  VESA1024.WGD   2.9k
WS.ICO        1.8k  WSINDEX.XCL    1.5k
```

```
Commands available from this screen
let you edit, print, and maintain
files and macros; use features such
as MailList, TelMerge, and Star
Exchange; and change your level of
displayed Help.

To see a menu, press a highlighted
letter in the menu bar. At a menu,
press the highlighted letter for the
command you want. Use the → key to
change menus. Press F1 for help.

You can also use classic commands
listed to the right of menu items.
To change to classic menus, select
Change Help Level at the Help menu.

To exit WordStar, press Esc, and
then X.

        Press Esc to continue
```

Figure 1. Word Perfect and AmiPro as Typical Examples of Text Processing Software in use During the 1990s.

Aspects of Workflow Automation

Instead of changing the original applications, the solution becomes the creation of an operating system and application-independent layer. The interaction could be handled in an abstract way by replacing the human user by a machine. This implies several tasks:

1. General high level control: Power on/off and reset the computer;
2. Reading and interpreting machine output from the screen;
3. Send keystrokes and/or mouse movements and clicks in interaction loops;
4. Inject and eject removable media like floppy disks or CD-Rom, DVD.

These abstract actions describe all machine and software environment-dependent workflows. The user chooses the appropriate operating system and/or application, selects the data carrier and object he wants to open, run or modify, and finally saves a certain state and shuts down the application or machine. As many migration workflows would imply an endless repetition of those tasks, they need to be automated in a suitable fashion in order to be deployable in preservation frameworks (von Suchodoletz et al., 2010). This could not be directly achieved on real, physical machines. But, the virtualization or emulation of the hardware would allow the replacement of removable media by image files, the screen frame buffer, keyboard and mouse by software devices accessible by other software, all using well-defined protocols. Thus, wrapped software environments become very versatile migration tools. The number of input and output formats supported depends only on the applications installed. This groundwork alone does not produce a concrete migration service. A further step would be to monitor and record a human user when executing a certain environment initialization or migration task. This recording is used in the future for repeated playback for the same task on the same type of objects. Similar recordings are required for each given input format generating a defined output.

So, the approach suggested here makes a technical and organizational separation between the machine used for workflows and the input/output (Figure 2). Emulated or virtualized environments are particularly well-suited for this. To define a valid migration service, deployable by a memory institution, the software environments and recordings need to be linked to other preservation services, described e.g., by the OAIS.

Interactive Preservation Workflows

Abstract human interaction is to be made operable by software through a one-time capture of the traditional interactive manner, which can thereafter be re-played (indefinitely). Such a setup requires an appropriate interface which can be used for computer-driven interaction.

VNC – The Ultimate Remote Control

The remote control software VNC is a very common solution to control machines remotely by sending keystrokes and mouse actions to a machine and copying the frame buffer content over the net. VNC builds upon RFB – a simple remote access protocol, which works at the frame-buffer level and is consequently platform independent.

A tool demonstrating this in the scope of the PLANETS project (Farquhar & Hockx-Yu, 2007) is GRATE. It is a prototype allowing for the wrapping of various software environments within a single application. It provides the archive user with an abstract interface independent of digital objects (Welte, 2009). GRATE uses VNC to reach an abstraction of a very wide range of different hardware architectures.

Vital parts of the GRATE functionality can be realised in a less complex way with the modular open source emulator QEMU[2]. It offers emulation of a wide range of hardware architectures, VNC support and hardware-monitor interfaces. Screen output and input via mouse or keyboard – which until now are still the most used methods of human-computer interaction – can be well simulated and observed using the standardized VNC protocol.

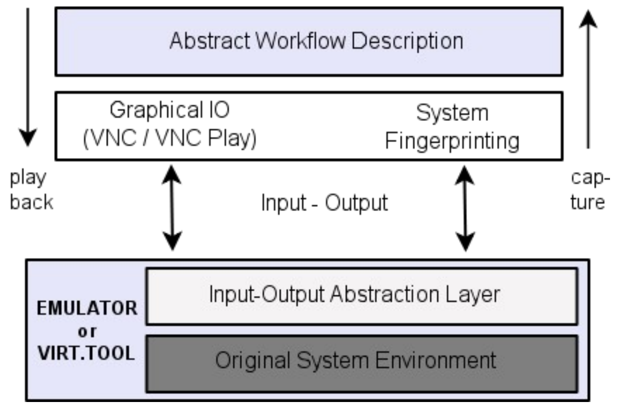


Figure 2. Abstract System Setup for Unattended Interaction Workflows.

The presented automation technique should be generalizable and testable with other suitable virtualization tools and emulators. The goal of future experiments is to prove it with another suitable emulator, like Dioscuri, which recently added a VNC interface (Genev, 2010).

Workflow Recording

The approach is to interactively record a particular workflow once, such as loading a deprecated Microsoft Word 1.1 document in its original environment and converting it, by printing with a suitable virtual printer, into a PostScript file. Such a recording can serve as a base for a deeper analysis and the generation of a machine script for subsequent, completely automated, repetition.

We define an interaction migration workflow as an ordered list of interactive events, which are passed on to the emulated environment through a defined interface. These events can be mouse or keyboard events, for example, but are not limited to these. Each of these events is linked with a precondition and an expected outcome, which can be observed as a state of the emulated environment. Until the expected outcome is observed, the next event cannot occur. Linking events with specific preconditions and outcomes is necessary since the workflow depends on the capacity of the emulation environment. Programs will take different amounts of time to run, depending on the load of the hosting machine. In the interactive case, this occurs through visual control of the user. For an automated operation, the definition of expected states and a reliable verification is indispensable.

[2] QEMU project homepage: http://www.qemu.org.

Synchronization and Machine Monitoring

In order to create interactive preservation workflows, the development of a reliable technique for producing and checking pre- and post conditions is necessary. Various technical approaches are required to record and verify the effects of interactive events. To observe the behaviour of the emulated hardware, (depending on the system), the screen output, the state of the emulated processor registers or a fingerprint of its main memory can be used to draw conclusions.

Following our experiments, we have found the approach of VNCPlay useful. This approach produces and compares snapshots of a small area around the mouse cursor for synchronization (Zeldovich & Chandra, 2005). This method was originally developed for the platform independent latency and general performance evaluation of graphical user interfaces. Nevertheless, a crucial factor is the handling of volatility. How much deviation from a certain state is allowed in order still to recognize it as the expected event? If the window placement algorithms do not produce exactly the same results, VNCPlay might wait indefinitely. The proper definition of thresholds is indispensable.

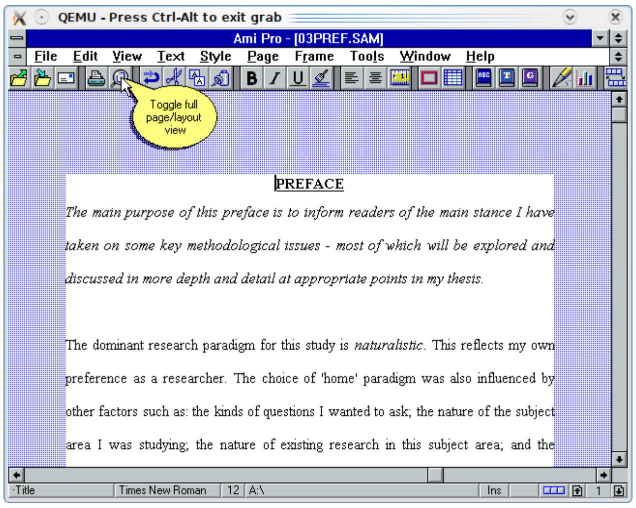
Moreover, in our setting we make use of an emulated environment, which not only allows the observation of external events of the running machine (e.g., screen), but also internal states of the emulated machine. However, most important is the ability to monitor and alter the state of peripheral devices in an automated way.

Example Workflows and Experiments

Given a certain primary object, an archivist creates a suitable environment the object could be rendered in. In the worst case, the process starts from scratch by installing the operating system on the emulator. In the best case, only the appropriate application has to be installed to a pre-existing emulator system image, if at all. Every transition between the software dependencies consists of recorded installation and configuration sessions within an emulated environment, possibly with some auxiliary data sources attached. This procedure could be conceptualized using view-paths, as explained later on. The selection of a proper environment could be automated by resolving a view-path, choosing the closest match of existing pathways in the software archive.

For example, the installation of the AmiPro application (Figure 3) requires a runtime environment consisting of a supported operating system and emulation of the necessary hardware components. In this case, QEMU was used to emulate a 386-compatible PC running Windows 3.11, based on MS-DOS 6.20 with CD-ROM support. In order to produce PostScript documents, an appropriate printer driver also has to be installed.

We do not record the installation process of operating systems, since these runnable images serve as end-points of every single view-path. Only transitions from these endpoints need to be captured. For this example, we recorded the installation of a PostScript printer driver and the AmiPro application as two distinct steps. For each process we rebooted the emulator with an on-the-fly generated ISO-9660 disk-image containing all necessary tools and installation files.


QEMU - Press Ctrl-Alt to exit grab
Ami Pro - [03PREF.SAM]
File Edit View Text Style Page Frame Tools Window Help
Toggle full page/layout view
Title
Times New Roman
12
A:\
Ins
1

PREFACE

The main purpose of this preface is to inform readers of the main stance I have taken on some key methodological issues - most of which will be explored and discussed in more depth and detail at appropriate points in my thesis.

The dominant research paradigm for this study is *naturalistic*. This reflects my own preference as a researcher. The choice of 'home' paradigm was also influenced by other factors such as: the kinds of questions I wanted to ask; the nature of the subject area I was studying; the nature of existing research in this subject area; and the

Figure 3. Lotus AmiPro Text Processor Executed in an Emulated Win3.11 Environment.

The result of both recordings is the full view-path for the designated file format of the primary object, and thus is ready for a view- or migration session. For both session types, the primary object has to be prepared for transportation into the emulated environment. For this example, we have created a floppy image-file containing the AmiPro (*.SAM) document.

Migration-sessions are designed to run in an unattended manner. Once the session is recorded it should be replayable on any document within the given specifications. The recorded workflow in this example was opening a SAM file with AmiPro and printing it with a PS-enabled printer driver directly to the attached floppy disk. We were able to successfully run a batch job doing a migration of a series of AmiPro documents. For each document, the image with the full view-path was freshly booted, a floppy image with the original file was created and the prerecorded workflow was executed. For each step the resulting PS-file was saved on the floppy and was available for further processing.

Results

	Scenario 1	Scenario 2
Real	4 min 28.804 sec	14 min 03.495 sec
User	0 min 04.652 sec	3 min 53.219 sec
Sys	0 min 01.420 sec	0 min 05.588 sec

Table 1. Playback Within Different Environments. Numbers created by the UNIX time command. Recording the workflow took 4 min 09.949 sec.

We have evaluated the playback under different environments and conditions (see Table 1). For example, the replay described in Scenario 1 was conducted under the same environment as that used for the recording. The elapsed time is comparable to the recording time. The small deviation is due to the extra time added after synchronization points to allow the operating system to be ready to handle the next input-event. In Scenario 2, the playback environment was under heavy IO- and CPU-load and therefore the playback took three times longer than the recording. Not only the total time increased (denoted as real), but also the CPU-time accounted to the process (denoted as user) increased significantly. This is due to more screen-fingerprint comparisons, while waiting for the matching synchronization point.

However, we have observed several failures while experimenting with different workflows and environmental settings. For example, creating the above-described view-path in the opposite order (first installing AmiPro and then the PostScript driver), the playback of the installation procedure failed at the very last step. The screen's fingerprint differed more than the allowed threshold of 5 % of mismatched pixels. This was mainly due to the change of the desktop background, as the icon of the previously installed printer-application was missing. Other failures were due to bad click or event (action, activity) timing. But with some training and more experience we were able to create working recordings with high success rates. Additionally, some extra measures can be taken to further improve the reliability. For example, the environment must always be entered and exited in a predictable way. There are two possibilities for entering a session: either the environment is booted or some (other) actions, like prerecorded tasks, must already have been executed. The system should look the same at any point in time, even if the session starts on top of different but compatible view-paths. Most importantly, all windows should be closed and if possible no background or minimized tasks should be running.

Whenever possible, keyboard-shortcuts should be used. They are particularly helpful if new windows are opened at a random position. With the appropriate key-combination, the window can be maximized and therefore the playback can always match the synchronization point. Furthermore, relying on a single screen-snapshot close to the mouse-pointer is not always sufficient. In contrast to the original purpose of the VNCPlay techniques for creating reliable and sustaining interactive workflows, some extra effort can be imposed on the recording user. We are currently developing a toolbox to alter and improve recordings by defining extra synchronization points and altering the designated area for fingerprinting but also choosing fallback strategies, e.g. if a mouse click was not properly recognized by the system.

Object Exchange Challenges

Beside the screen-keyboard-mouse interaction, the attachment (and detachment) of the primary objects is a major part of any automated processing setup (Figure 4): The objects need to be passed in and out of the emulated or virtualized environment. Several different strategies could be followed:

- Work with a single, fixed or variable number of objects per session, defined as the timespan between start and shutdown of a certain environment;
- Load the objects before a session starts and unload them after the end or while the session is running.

This decision is not a trivial task and depends on the feature set of the original environment. A major challenge is the tight interlinkage with the workflow recording. Defined reference points are needed in order to define where to start from, e.g. the name(s) of the source file(s). If the filename and, in the case of graphical environments, its displayed position on the graphical user interface, is not nearly enough the same as during the workflow recording, the playback will most likely fail as the expected state never occurs. This complicates the task of working on multiple files without changing the removable media between each one.

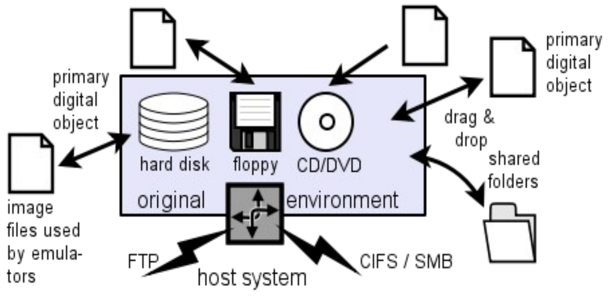


Figure 4. Depending on the Recreated Environment and Type of Emulation, Several Options Exist for the Transport of the Digital Objects.

ISO and Floppy Disk Images

Both devices are typically removable and thus offer a data exchange option while the emulator or virtualization tool is running. The emulator must support virtual media loading and ejection functionality, otherwise media changes might not be noticed. Not all hardware platforms and operating systems support optical drives (e.g., ISO-9660 images), but most support floppy disks. Floppy disk drives often do not require special driver setup besides the built-in operating system capabilities, which is different from optical drives. The (un)loading of removable media requires an additional channel beside VNC, which is used for playback. This channel needs to be synchronized with the workflow recording and playback, adding complexity to the whole setup.

System Images and Container Files

Emulators usually use so-called container files as virtual hard-disk images containing the installed software environment. Therefore, they offer an option to transport a digital object into the emulated environment by embedding it in the container file, or by creating a secondary one, which is then attached as an additional virtual hard disk. The hardware drivers needed to handle hard disks are part of most operating systems and thus a minor issue to be considered (Figure 5).

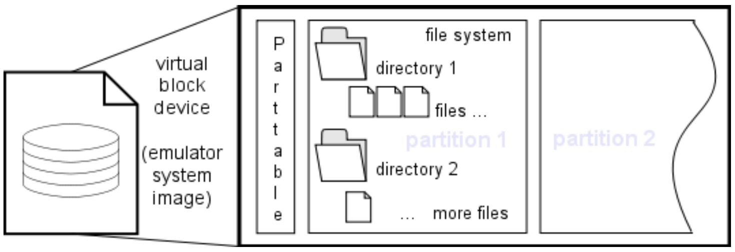


Figure 5. Emulator System Images are Block Devices Containing the Original Environment in an Emulator or Virtualization Tool Specific Format.

Network-Transport

Many of the newer and advanced operating systems offer low level access to standard network interfaces. Such interfaces were widely integrated in the early eighties, but first into the high-end commercial systems. In the mid-nineties a network interface became standard on every desktop machine. In order to use these interfaces it often required the installation of additional software, like hardware drivers and network protocol stacks. In the beginning, there was a wider range of higher level network protocols in use, but in the nineties the Internet Protocol (IP) became more dominant and soon a quasi-standard. Support for the various protocols were usually not part of the standard features of deprecated operating systems. Thus, additional software had to be purchased and installed separately. These software packages have to be conserved in the software archive too.

The main advantage of network-transport is the synchronous operation: while running the reference environment, any exchange of objects is possible in both directions. The size of files is only bound to limitations of the deployed operating system. The File Transfer Protocol (FTP) is one of the oldest protocols using TCP/IP. It has been around for more than 30 years and has not changed very much. In the early days of FTP, there were only simple command line tools with a small footprint. Now, most of the modern operating systems or additional tools implement comfortable front ends to this protocol. The NCSA telnet package contains an FTP client, as does every Windows version from 98 on. It was standard for the different Unixes from nearly the beginning.

The SMB (Server Message Block) protocol, and its successor CIFS (Common Internet File System) are not just simple file transport protocols, but also network file systems. Thus, they offer more convenient transportation of files back and forth and, furthermore, these features are directly integrated into the standard file system of the running operating systems. SMB was originally invented by IBM with the aim of turning the DOS Interrupt 33 (21h) local file-access into a networked file-system. Later on, Microsoft made considerable changes to it. SMB has been available in Windows since the Workstation release of 3.11 and Windows NT. It is also implemented for the different Unixes and similar systems, and known as the Samba software package[3]. The existence of many implementations of SMB/CIFS, especially the Open Source Samba package, should offer a long time support for that protocol well after its demise.

Considerations and Findings

The main advantage of virtual floppy disks is the ease of use: the support of them by emulators and virtualization tools is very good and additional drivers are not required within the operating systems. The downside is their size limitation: only a few and rather small number of files could be added to them. ISO images offer much larger capacity but are by definition read-only. Thus, it is impossible to use them for passing back modified objects. Plus, they typically require a more complex driver setup compared to floppy disks.

If the number of objects or single objects are larger, then larger structures than floppy disks are needed. Here the virtual hard disks could solve the issue. However, for producing or modifying such containers, exact knowledge of the internal format and the file system is required. Furthermore, as hard disks are typically not hot-pluggable, the object(s) have to be attached before the emulator or virtualization tool is started. Plus, the modified version of the digital object(s) is only available after the emulator has been stopped. Otherwise any changes to the block-layer might corrupt the container.

[3] The Samba Project: http://www.samba.org.

In order to pass a larger number of objects, network file transfers or file systems may be used as an alternative. An additional control channel to the virtual hardware is not required. Nevertheless, the provisioning of the objects needs to be synchronized with workflow recordings and playbacks. Networking is not available for all relevant environments and definitely needs a more complex driver setup. Modern virtualization tools offer additional ways of data exchange – shared folders, directories shared between the host and the original environment, and "drag&drop" are available for newer software environments, as special drivers are now required.

To simplify the setup in all experiments conducted up to now, the objects were packed and attached onto virtual floppy disks before the emulator was started and extracted after shutdown to avoid the handling of additional channels.

View-Paths as Tool Chain Formalization

Up to now pre-configured environments were presumed. This is acceptable for a few experimental migration tasks, but not realistic for larger memory institutions with a wide range of different objects. Each type of digital object requires a suitable context in order to be accessible. This context must combine hardware and software components in order to create the original or a compatible environment where the object can be reliably processed. The procedure could be formalised by so-called "view-paths" or pathways. Those are directions from the primary object of interest into the actual environment of the user (von Suchodoletz & van der Hoeven, 2009; von Suchodoletz, 2009). A view-path delivers a vector originating from the digital object to its creation application, the required operating system and the resulting hardware emulator (Figure 6). The additional components needed in this process are called secondary objects.

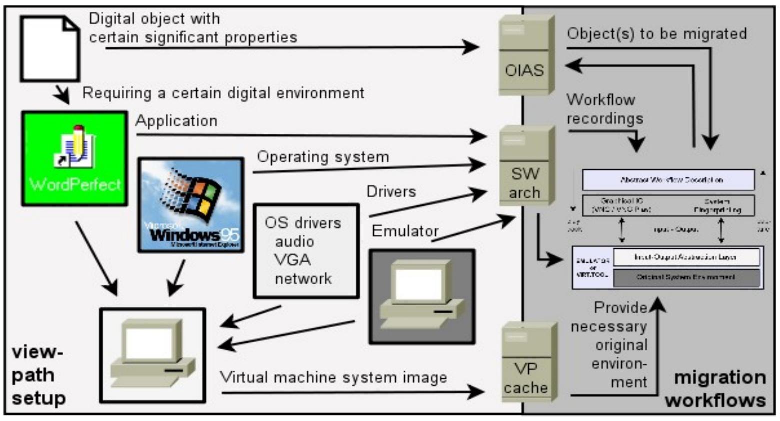

Digital object with certain significant properties
Requiring a certain digital environment
Application
WordPerfect
Windows 95
Operating system
Drivers
OS drivers audio VGA network
Emulator
Virtual machine system image
view-path setup
OIAS
SW arch
VP cache
Object(s) to be migrated
Workflow recordings
Abstract Workflow Description
Graphical IO (VNC / VNC Plus)
System Fingerprinting
Input - Output
Input-Output Abstraction Layer
EMULATOR or VIRT.TOOL
Original System Environment
Provide necessary original environ-ment
migration workflows

Figure 6. View-Path Describing How the Original Environments are to be Created for Use in Migration Workflows.

There are two end-points determining a valid view-path: the primary object and a suitable emulated system acting as bridge to the archive users' current working environment. First, the digital object has to be characterised in order to determine whether and which application is needed for rendering. Then, depending on the object and the application, the operating system or necessary level of hardware-emulation, a range of additional software needs to be taken into account. This may be helper programs, fonts codecs or drivers. The vector should finally point to a suitable emulator for the given actual environment. As a result, there might be different view-paths for each object-type, because usually there is more than one rendering application or system emulation available. Plus, the number of view-paths to be managed by the archive increases with every new primary object type added. To actually produce a chosen view-path extra to the software itself, contextual information like installation manuals or license keys are required. Unfortunately, a view-path does not directly deliver a software installation plan and thus could only partly be used to automate setup workflows.

To support the realization of a view-path for a primary object, we need to manage the preservation of the required secondary objects. This is the role of the software archive (von Suchodoletz et al., 2010). Because certain object types have a relatively high complexity, some issues need to be considered about how a view-path is computed. Especially for frequently requested view-paths, it could be conceivable to work with prepared, cached environments. Altogether, this leads to the following requirements for managing the software archive:

- **Creation of a Software Archive:** Every single object needed to create a certain view-path is permanently stored within the archive. These additional objects are to be kept as the primary objects of interest. At this point it could be considered whether the view-path objects, such as emulators, operating systems, specific helper software and description, are bundled together in a single package or stored individually.
- **Operation of an online archive for fast access:** For frequently requested secondary objects, it is more efficient to store these in a special archive, additional to the long-term archive, both to reduce the load on the long-term archive and to improve the process of generating arbitrary view-paths.
- **Setup of a view-path cache:** For often requested and more complex view-paths, the use of a prepared working environment can reduce the work for users and archive operators. This cache is either part of the online archive or is directly available on the reference platform. A major challenge archivists face is the interactive character of most of the software components involved. Hence, some steps need to be executed through direct machine interaction. Once the view-path is created for rendering primary objects of a certain type, it could be reused on objects of the same type.

Optimization and Future Research

Especially for large-scale migration scenarios, the performance of each single migration step is relevant. Optimizations are possible on different levels. The migration automation approach demonstrated is completely agnostic of the underlying operating system and deployed applications. Thus, we ignored some of the trivial workflow automating achievable by using the autostart mechanism of operating systems. If the application is started on file load (by `<application> <filename>`), further reduction of interaction would be possible.

Woods and Brown (2010) suggest scripted on-demand installations of common software environments in their "Assisted Emulation" paper. The authors additionally advocate to preserve necessary contextual information through scripts designed to control the legacy environment, and created during the preservation workflow.

Ad-Hoc View-Path Generation Versus Prefabricated View-Path Archive

In the moment of disseminating a primary object from the archive for object rendering, it needs to be processed and then executed, automatically or with the interaction of the user. These work steps imply not only copying and reproducing the object's bit stream, but actually providing access to the object in a sensible way. At this point, emulation and migration strategies do not differ much: the procedures for the object reproduction could both be described by the aforementioned view-paths.

For the actual deployment of view-paths, mainly two different approaches could be identified:

- Generating the software stack from scratch directly from the components stored in the software archive.
- Using prefabricated setups generated once by an archivist and then stored as an emulator container file in the view-path cache of the archive.

Of course, any variant in between would be possible too. While the first option would require lots of repeated manual work or automation (e.g., using the approach presented in this paper) it would reduce intermediary items to be stored and updated if the emulation environment changes. The latter reduces the delivery time and complexity of often-requested view-paths.

Each item of the view-path cache needs to be described in the software object repository to be chosen accordingly and compared to the multi-item on demand view-path setup. In any case, the archive user needs a certain set of tools to access the object.

Workflow Optimizations

The experiments described in this paper relied solely on virtual floppy disks for the object transport with single objects embedded, which were attached before the start of the emulator and extracted afterwards. Disk image could override limitations in size and number of processed objects. This would require more research into how to atomize workflows properly to process an arbitrary number of objects consecutively. Other ways of object transport, like network filesystems, would allow synchronous access to the objects to be migrated without starting and stopping the emulation.

The results are to be verified in different setups and the performance of different workflows must be measured in order to generalize, for example, by using Dioscuri as an alternative emulator. Measurements should be discussed and then applied for a more general evaluation of the different workflows.

Conclusion and Outlook

The suggested novel approach allows the migration of virtually every type of digital object once created in a typical desktop user environment. By using the original applications or environments, it circumvents the problem of non-existing migration tools. Nevertheless, the possible output formats are limited to the types programmed in to the specific application or available within the original environment.

With the described techniques, tools and experiments we have shown, our approach is a promising and suitable way to construct workflows for instantiating view-paths, preparing view sessions and executing interactive migrations, all in an unattended way. The advantages of an automated wrapping of interactive environments are twofold. It is possible to run large batches of jobs in an unattended fashion. Also, untrained people, such as the average archive users of the future, can access ancient environments without knowing exactly how they have to be handled. For such view-sessions, only a few automated actions need to be recorded in order to present the user with the content of a deprecated text document. Thus, for more complex applications on older GUI environments, preparing a ready-made view-session may increase productivity of the archive users. Plus, workflow recordings might provide enhanced software documentation as significant actions made available for playback.

Nevertheless, emulation is a strategy with certain complexities, requiring specially trained archivists. Aside from the emulated computer environment, documentation and skills are needed to understand how to operate an old computer environment in order to set up the required view-path. As of today, many of us still remember older environments, such as MS-DOS and early Windows versions, but soon even that information will disappear as much of the mainframe operation knowledge already has. Therefore, manuals, tutorials and practical how-to guides also need to be available. The emulation software archive needs to contain not only the emulators and all components for the emulated environments, but has to be supplemented with a new section, namely the interactive workflows and associated view-paths.

However, for unattended interactive migration tasks, some important challenges remain. Most importantly, such recorded workflows should be described on an abstract level and should therefore be editable. Hence, such abstract workflows can be treated as digital objects with possible migration paths to facilitate future compatibility. Similarly, the description format should support branches in order to allow different behavior in different environments. If, for example, an action fails, in some cases a rollback and/or retry from or to a defined checkpoint should be possible. Thus, a higher probability of successfully applying a prerecorded workflow in a slightly different environment could be achieved.

Usability and reliability can also be improved by providing appropriate tools. Editing recordings allows the user to improve recordings, alter or set synchronization points manually and to select a strategy for failed actions. Also, enriching recordings with descriptive meta-data will help future users understand the recorded workflows.

Disclaimer

This article is based on a talk presented at the iPRES 2009 conference in San Francisco (Rechert et al., 2009), the results accomplished during the PLANETS project and the ongoing research in the Open Planets Foundation[4]. Part of this work was supported by the European Union in the 6th Framework Programme, IST, through the PLANETS project[5], contract 033789. The authors are solely responsible for the content of this paper. It does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of data appearing therein.

[4] OPF, started in May 2010, is the official PLANETS follow up project, see http://www.openplanetsfoundation.org.

[5] PLANETS Project: http://www.planets-project.eu.

References

Farquhar, A., & Hockx-Yu, H. (2007). Planets: Integrated services for digital preservation. *International Journal of Digital Curation, 2, (2).*

Genev, E. (2010). *VNC-Interface for Java X86-Emulator Dioscuri.* Retrieved February 15, 2011, from http://eprints.rclis.org/19352.

Mellor, P., Wheatley, P., & Sergeant, D. (2002) Migration on request, a practical technique for preservation. In G. Goos, J. Hartmanis, & J. van Leeuwen, (Eds) *ECDL 2002 Vol. 2458*. Berlin: Springer.

van der Hoeven, J.; van Diessen, R.; & van der Meer, K. (2005). Development of a universal virtual computer (UVC) for long-term preservation of digital objects. *Journal of Information Science 31, (3).*

Rechert, K., von Suchodoletz, D., Welte, R., van der Dobbelsteen, M. Roberts, B., Schroder, J., & van der Hoeven, J. (2009). Novel workflows for abstract handling of complex interaction processes in digital preservation. *In Proceedings of iPRES 2009.* San Francisco, CA, USA: California Digital Library.

Rechert, K., von Suchodoletz, D., & Welte, R. (2010) Emulation based services in digital preservation. *In JCDL '10: Proceedings of the 10th annual joint conference on digital libraries.* Glasgow, UK.

Tzitzikas, Y. (2007) Dependency management for the preservation of digital information. *Lecture Notes in Computer Science.* Berlin: Springer.

von Suchodoletz, D. (2009). *Requirements for emulation as a long-term preservation strategy*. Retrieved February 15, 2011, from http://eprints.rclis.org/18984.

von Suchodoletz, D., van den Dobbelsteen, M., & Rechert, K. (2010). *Software archives as a vital base for digital preservation strategies.* Retrieved February 15, 2011, from http://eprints.rclis.org/18764.

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

Welte, R. (2009). *Functional long-term preservation of digital objects - development of a demonstrator for the Internet: Use of emulated execution environments.* Retrieved February 24, 2011, from http://www.freidok.uni-freiburg.de/volltexte/6605.

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

Zeldovich, N., & Chandra, R. (2005). Interactive performance measurement with VNCplay. *In ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference.* Berkeley, CA, USA: USENIX Association.