Replicating Installed Application and Information Environments onto Emulated or Virtualized Hardware

Dirk von Suchodoletz
Institute of Computer Science, Albert-Ludwigs University
10 H.-Herder st., Freiburg, 79104, Germany.
dirk.von.suchodoletz@uni-freiburg.de

Euan Cochrane
Archives New Zealand, The Department of Internal Affairs
10 Mulgrave st, Wellington, 6011, New Zealand.
euan.cochrane@dia.govt.nz

ABSTRACT

Digital objects are often more complex than their common perception as individual files or small sets of files. Standard digital preservation methods can lose important parts of digital objects, or the context of digital objects. To deal with the different types of complex digital objects, and to cope with their special requirements, we propose applying emulation from a different perspective in order to preserve the whole original environment of single digital objects or groups of digital objects. Many of today's preservation scenarios would benefit from a change in our understanding of digital objects. Our understanding should be shifted up from the single digital files or small groups of files as they are commonly conceived of, to full computer systems. When this shift in perspective is undertaken two important outcomes result: 1. the subject of preservation includes a much richer level of context and 2. the tools available for preserving them are constricted. In this paper we describe a workflow to be used for replicating installed application environments that have the x86 architecture onto emulated or virtualized hardware, we discuss the potential for automating steps in the workflow and conclude by addressing some of the possible issues with this approach.

Keywords

Emulation, Disk Imaging, Virtualization, Complex Digital Object, Original Digital Ecosystem

1. INTRODUCTION

Most digital objects are more complex than perceived by archivists or other practitioners dealing with the task of digitally preserving office workflows, scientific desktops, electronic publications or dynamic objects like multimedia encyclopedias, educational software and computer games. The majority of today's digital objects consist of individual files but most of those files are not self contained. A digital ecosystem is required to render or run these digital objects. In order to preserve the individual files, and the entirety of the information that is presented when they are rendered, it would be useful, and in many cases necessary, to be able to replicate and preserve their original rendering environments. Overall there are at least three compelling reasons for making images of entire information environments and maintaining the ability to render them over time:

1. To provide researchers the ability to experience individual users' or representative users' old information environments such as politicians', artists' and other famous peoples' information environments or an average/representative user's Information Environment from a particular time period.
2. In order to preserve complex digital objects in an inexpensive and efficient way by enabling the automation of their preservation.
3. To produce permanent "viewers" for digital objects that can easily be maintained over the long term and are known to be compatible with the objects.

Preserving a famous person's installed application and information environment has been demonstrated most successfully by the team at the University of Emory's Manuscript Archives and Rare Book Library (MARBL) where they have preserved an image of the hard disk from Salman Rushdie's early 1990s Macintosh desktop and currently use an emulator to access it [6]. The value of this approach has been realized by the team at Emory where they have seen new ways of conducting research being developed because of the availability of the emulated desktop. The experience of interacting with the original owner's actual desktop environment has led to novel discoveries being made such as the discovery the Emory team made that Rushdie was an avid user of the "stickies" application on the Mac operating system.

Unfortunately, imaging and maintaining access to old computer desktops is still a niche endeavor for a number of reasons including the perceived complexity and difficulty for average, non-technically trained preservation practitioners. This need not be the case. The steps described in this paper constitutes a feasible workflow that suitably skilled practitioners could use immediately to begin replicating installed application and information environments onto emulated or virtualized hardware. The workflow also includes numerous steps which have the potential to be highly automated, making the process easily manageable by an average archivist,

iPRES2011, Nov. 1–4, 2011, Singapore.

librarian or other less technically skilled digital preservation practitioner.

2. PRESERVING COMPLEX OBJECTS

Digital objects are often assumed to be individual computer files that stand-alone and don't have many dependencies aside from requiring some sort of program to render or "open" them.

This understanding of digital objects is very limited and quickly breaks down under closer examination. For example, there are many types of digital objects that include more than one "content files" such as digital videos which are often captured as thousands of image files with a separate audio file and separate metadata file, or linked spreadsheet-workbooks in which one table provides data to another. In examples like this the loss of any one file from the digital object can lead to the inability to render the wider object [4].

As another example, files stored in Electronic Document and Records Management Systems (EDRMS) can be ruined from an archival perspective if they are taken out of their EDRMS as their context can be lost. In this example it would theoretically be possible to define a metadata standard and preserve sufficient metadata to capture the context that the file came from; however in practice this is extremely difficult. A further example is provided by the case of office documents (e.g. Microsoft Word) that require additional/special fonts to be rendered. Without the fonts these digital objects sometimes cannot be rendered at all [1, 9]. These examples illustrate that digital objects are often more complex than they are generally conceived. In addition to this, when the necessary additional files are not saved directly with the "content files", it is often unclear and difficult to ascertain what additional components are necessary for rendering the object with integrity. This presents a real and significant problem to archivists, librarians, curators and other digital preservation practitioners who are faced with the task of preserving such objects.

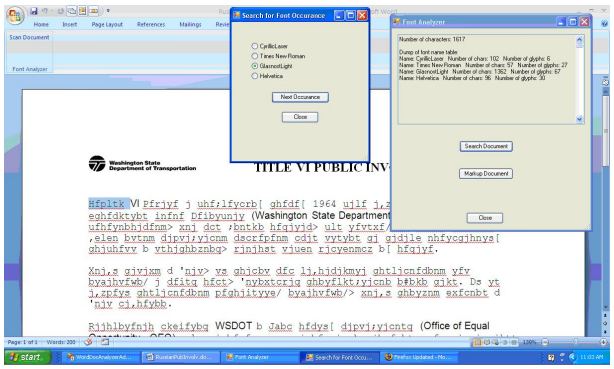

Figure 1: Missing fonts can make digital objects unintelligible [1].

All of these cases benefit from a shift of our understanding of digital objects up from the single digital files or small groups of files as they are currently conceived of, to full computer systems. By shifting our understanding up to this level we enable conversations and planning to be undertaken to identify such objects. This shift in understanding also enables preservation institutions and research organisations to begin to address the practical preservation requirements that need to be fulfilled in order to preserve such complex objects for future generations. The experience of maintaining access to old software like computer games using emulation [2, 8] helps to understand the envisioned process.

The workflow described in this paper provides one initial practical option for preserving such complex objects. The workflow can be used to preserve the complete creating and/-or rendering environment portion of a complex digital object, ensuring that any dependencies are preserved. This approach would significantly help in solving the problem outlined above.

3. PRODUCING PERMANENT, COMPATIBILITY VERIFIED VIEWERS

When addressing the problem of long term access to (or preservation of) digital objects there are two primary options that preservation practitioners have for solving it:

1. Move the information from one file or set of files to another file or set of files which can be rendered more easily using existing software.
2. Maintain the original rendering software indefinitely.

Option (1) implies many potential problems including difficulty in ascertaining whether migrated versions of objects have retained their integrity, either due to information loss in the conversion process or difficulty in assessing whether new renderers are truly compatible with the objects, and a potentially high long-term cost due to having to perform migrations on a regular (if infrequent) basis. Option (2) has been considered unfeasible for various reasons including (though not limited to) the difficulty in maintaining the viewers over time.

The workflow outlined in this paper could be used to produce viewing environments that would be known to be compatible with an organization's entire set of digital objects or digital objects across a time period. This would solve one of the problems faced by option (1) by providing a compatibility verified viewer for the objects. For example, a standard desktop environment from a public-sector organization that is intended to be able to be used to view/render all the objects created by that organization could be replicated onto virtual or emulated hardware, or a web browsing environment representing a period of time could be replicated and maintained for use indefinitely in viewing/rendering websites from that period.

Virtualized or emulated environments are designed to be portable across different types of hardware and operating systems, including potential future hardware and operating systems, and thus viewing environments created using the workflow outlined in this paper would have these same sustainability properties. Furthermore, while longevity is a general property of virtualized or emulated environments, the workflow outlined in this paper could also be used to

replicate environments for use in emulators that have been specifically designed to need very little maintenance over time, such as the Dioscuri modular emulator. This would ensure that there were even more sustainable and viable as permanent solutions. Virtualization products may be seen as being quite short-term solutions from a digital preservation perspective however they are are practical in so much as they are available now and can be used to solve current problems. Furthermore virtualization products often use emulation to provide many critical components of their total product and are best seen as emulators (e.g. VirtualBox).

4. ENVIRONMENT REPLICATION WORK FLOW

After some initial hard drive imaging and virtualization tests with a MySQL database system running on Linux to investigate the general feasibility and practicality in 2007[1] the authors undertook more elaborate and broader experiments at Archives New Zealand. In these activities we aimed to investigate the feasibility of making images of old computers that were running a wide range of DOS and Windows operating systems. The experiments aimed to replicate the installed application and information environments on the computers' disks onto emulated and virtualized hardware. This imitated the idea of moving a hard disk from one physical machine to another compatible machine, and thereby preserving most (if not all) relevant aspects of the first one. This was achievable as almost all the relevant information on a computer is kept on its hard disk. In some circumstances this approach may not be feasible due to some components being provided through external hardware or some aspects of the environment being dependant on particular firmware code, however for the most part this approach is very effective. The experiments were very successful and from these experiments a number of steps were documented and have been formed into a workflow. The workflow outlines how to replicate installed application and information environments onto emulated or virtualized hardware. A detailed discussion of the experiments, including additional findings, will be included in a forthcoming paper. The following section describes the steps involved in the workflow and outlines the options and decision points that occur within it.

4.1 Create Disk Images

The first step in the workflow is to make images of the hard drive or drives that contain the environment that you wish to replicate in emulated or virtualized hardware. There are two ways of doing this: It can be done intrusively by taking the hard drive out of the old hardware and attaching it to modern hardware, or non-intrusively, by running modern software in the memory on the old hardware and making an image of the drive(s) over a network connection.

Intrusive Disk Imaging. Intrusive disk imaging involves removing the target hard drive(s) from the original computer hardware. For this reason is both difficult for someone without specialist training to undertake and potentially quite risky as the hardware may be damaged during the process. There are two initial steps involved in this method:

1. Detach and remove the hard disk from the computer case.

2. If the connector is IDE or SATA there are then two secondary options:

   (a) Connect the hard disk to an USB-IDE/SCSI adapter and plug this one into a machine running Linux that the image is to be "dumped" (written) to.

   (b) Directly connect the drive to a machine that has a compatible IDE/SATA/SCSI connector on its motherboard or has an appropriate extension card installed and can run Linux.

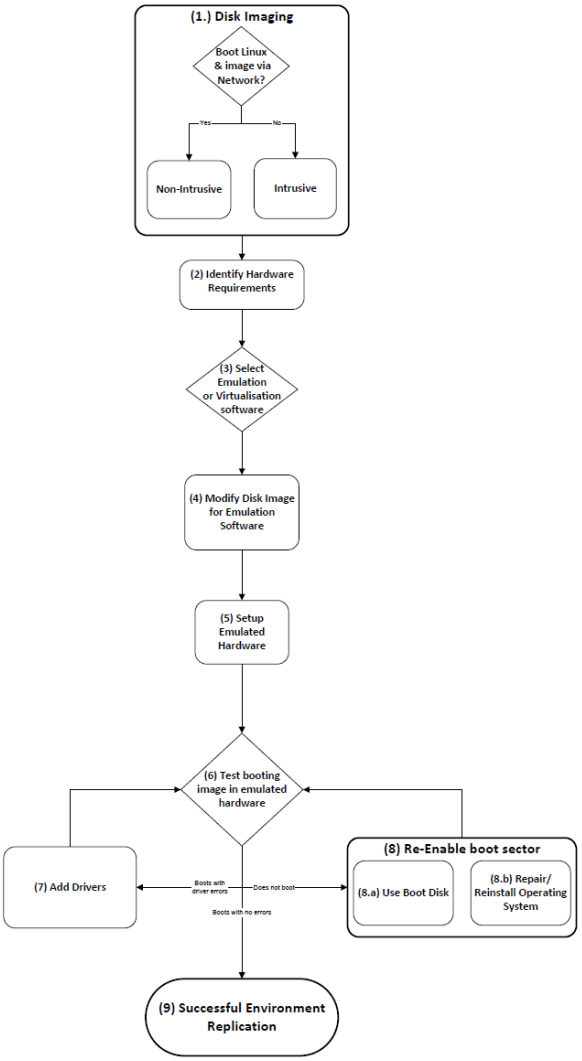

Figure 2: Formalized work flow diagram

[1] An x86 IBM server machine containing a hardware raid of three SCSI disks was dumped running in single user mode with network connection and file system set to read only. The resulting image was converted to VMware image type and the hardware driver changed to Buslogic [13, p. 165].

A computer may need to be switched off to attach or remove a disk so is a good idea to have a dedicated computer to do this with in order to avoid complications. Option 2a may be preferred in many cases as it is easier due to not having to open up the modern hardware in order to connect the old hard drive to it. However it may be substantially slower than option 2b which may be important in some contexts.

Copying the content. Once the disk is connected to a machine running Linux it is then possible to run a command to copy the entire contents and structure of the hard disk into an image file. In Linux the program that performs the imaging procedure is `dd`. To initiate the process the full command used is as follows: `dd if=/dev/sdb of=imagename.img` Where `sdb` is the logical device identifier of the attached disk in the Linux system. The disks are enumerated starting from letter a. Thus additionally attached disks are labeled in alphabetical order. Using the `fdisk -l /dev/sdb` command helps to identify the disk by its size if several are attached.

The `dd` tool reads the disk contents directly from the device blockwise from the absolute beginning to the end and thus requires administrator permissions in order to be executed. Directing `dd` to image `/dev/sdb` is the simplest imaging method as it creates a copy of the entire disk and any partitions on it. However it is possible to just make an image of the content of any relevant partitions such as imaging the second partition with `/dev/sdb2` to save on size of the resulting disk image. In the latter case the boot sector and partition table are lost though, as the disk layout is changed from a partitioned one to a linearly used disk without partitioning on it. This in turn makes running the image in an emulator much more complicated.

If `dd` encounters problems because of bad sectors then the `dd_rescue` program can be used as an alternative to help to produce an image from the readable sectors. The resulting image will be usable in many cases provided the sectors containing significant portions of the boot sector or operating system have not been affected.[2]

The time taken to image a disk using this method will vary depending on the type of connection, the size and the speed of the disk being imaged. It can take from less than one minute to more than an hour for very large disks. The process is reasonably straight forward for standard disk configurations such as those that are likely to be found in standard desktop office machines. It becomes more challenging when involving hardware or software RAID setups which might be present in servers. In these cases it might be possible to access the disks via non-intrusive imaging. Furthermore this process will likely be greatly simplified for many current day servers which are often already running from virtual disk images as virtual machines.

Non-Intrusive Disk Imaging. The least intrusive disk imaging method is to boot a "live" distribution of the Linux operating system into the memory of the computer, and image the hard disk over a network connection directly to another machine (Fig. 3). These "live" distributions do not use the internal hard disk and thus preserve its original state and leave it free (not in use) to be imaged. The "live" distributions come in a number of forms that can be run on varied sets of hardware. Linux includes a wide range of popular hardware drivers. Support for standard IDE and SCSI disks will be included in most distributions; many hardware RAID controllers are also supported. Most versions of "live" Linux distributions consist of a CD-ROM which can be booted from. It is also possible to find distributions that include a boot-floppy disk for machines that require this to be run first in order to boot from the CD-ROM. For machines that can boot from USB, Linux distributions that run from USB drives can be found. If the machine is an older one without a CD-ROM drive, a compact Linux distribution can be used that will run from one or more floppy disks such as MuLinux.[3]

Figure 3: Booting Damn Small Linux on an x86 Compaq desktop machine to allow non-intrusive disk dumping

Once an appropriate Linux distribution has been selected it needs to be booted in "live" mode on the original hardware to a point where the command line is accessible. The hardware needs to be connected to a *dumping* target machine via a network connection and the connection has to be established between the two.[4] The Linux commands `ifconfig` or `ip` can be used to setup the connection, assign an IP address to the hardware containing the target hard disk. The connection can be tested using the `ping` command. If there is no Linux hardware support available for a given disk configuration it

[2]The `dd_rescue` program, unlike the `dd` program, is not normally included by default in Linux distributions.

[3]See: *http://www.micheleandreoli.it/mulinux*

[4]We keep a selection of Linux supported NICs: PCI, Cardbus, ISA to install them into the target if required.

Tool / Image type	dd-raw	vmdk	vdi
QEMU	x	x	x
VirtualBox	–	x	x
VMware	–	x	–

Table 1: Compatibility list of container formats understood by the different x86 virtualization tools or emulators.

would be possible to find other disk imaging tools which produce similar results.[5]

Copying the Content. As when imaging a hard drive that is directly connected to modern hardware, the command to use on the just started Linux is `dd`. To initiate the process the full command used for the machines that is to send the image, via SSH (a secure data exchange protocol), to the image storage computer over the network will be similar to this:

```
dd if=/dev/hda|ssh username@ip.of.target.machine
dd of=imagename.img
```

Where the *username* was the username to be given for the remote machine, the *ip.address.of.target.machine* is the IP address of the computer the image data is being sent to, and the *imagename.img* is the name that was to be given to the file to that the image was to be written to. Some older Linux kernels might use different naming for IDE disks like *hda* instead of *sda*. `fdisk -l` usually gives the information on the installed disks of a computer.

The time taken for this process will vary significantly depending on the throughput of the network connection and the size of the hard disk being imaged, from minutes to hours.

4.2 Emulation Software Preparation

Once a copy of the original hard disk has been written to an image file the steps to resurrect the environment in emulated or virtualized hardware can be begun. The first step in this sub-process is to identify the hardware requirements of the software environment that is being replicated. In order to select an appropriate emulation or virtualization application it is first necessary to identify the hardware requirements of the software environment to be replicated so that they can be correlated to those provided by the different emulation and virtualization applications. The choice of software to be used to provide the virtualized or emulated hardware to replicate the imaged application and information environment on will depend on a number of factors.

Hardware Provision. Availability of necessary virtual or emulated hardware is one of the primary considerations when selecting a virtualization or emulation application for long term preservation/access purposes. For example, if the environment being replicated is to be used to access networked resources such as old websites, then it will be necessary for the emulation or virtualization application to provide a virtual or emulated network card that is compatible with the operating system of the environment that is being replicated. Depending on the virtual/emulated machine, numerous hardware configuration options are available:

- QEMU[6] supports numerous different graphic cards like Cirrus gl5446, generic VESA or the VMware SVGA II and different sound and network adapters (ne2000, AMD PCnet, rtl8029, rtl8139, Intel e1000, ...)
- Virtual PC offers S3 Trio32/64 graphics adapter
- VMware Workstation provides an SVGA adapter without a real world counterpart and different kind of network adapters: the AMD PCnet, the Intel e1000 and virtual IO.
- Virtual Box is flexible regarding the chip set (PIIX3,4 and ICH) and other options like Soundblaster 16 and ICH AC97 audio.
- DosBox[7] implements S3, et3000, et4000 and other graphics adapters, various sound cards and a higher layer networking.

Operating Systems and Hardware Drivers. Hardware and operating systems were once tightly matched for a number of older computer platforms such Motorola or PowerPC0-based Macintoshes or Atari home computers. This was not the case the with x86 architecture ("PCs"). The system BIOS of modern computers provides a number of standard APIs for the operating system to access the hard disk, floppy drive and graphic adapter. The operating system only needed specific hardware drivers to exploit the full feature set of other components like network and audio adapters or a graphic card's 3D capabilities.

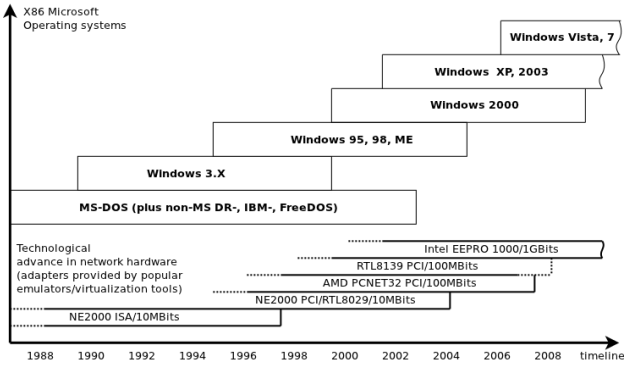

Figure 4: Driver support in standard x86 operating systems compared to provided emulated network adapters

[5]Open Source solutions like Clonezilla, *http://clonezilla.org* or commercial products Norton or Symantec Ghost and alike

[6]A popular Open Source multi-platform emulator for X86, PowerPC, ARM, S390 and others, see *http://wiki.qemu.org*. See for DP considerations [14].

[7]Open Source x86 and DOS emulator for BeOS, Linux, Mac OS X, OS/2, and Windows, *http://www.dosbox.com*.

Unfortunately, the driver situation of commercial operating systems for older x86 architecture computers is rather diverse [13, p. 190-195]. There is no driver support in DOS or early Windows versions for modern hardware. The emulated or virtualized hardware set must match the era the operating system was available and so emulators have to provide a wide range of different hardware configurations if they are to be able to host the x86 operating systems covering more then two decades. (Fig. 4).

Tool Considerations. Another major consideration is longevity. Software that provides emulated hardware is the only viable option for digital preservation as emulating hardware does not require running the emulator application on hardware that is compatible with the emulated hardware. Virtualization, on the other hand, relies (for the most part) on the underlying hardware, which the software that provides the virtualized hardware runs on, to be compatible with the virtualized hardware (e.g. to run an application that provides virtualized PowerPC hardware the underlying hardware on which the application is run must also be PowerPC compatible). A further consideration is licensing cost: Those vary amongst virtualization and emulation vendors and over their numerous products and services, and in some cases this may restrict available options. However in other cases it may be advantageous for institutions that already use particular virtualization software to continue to use this software for their digital preservation needs over the medium term (over the long term emulation is the only option as identified above).

Modify Disk Image Format for Emulation or Virtualization Software. The QEMU tool suite provides a tool "qemu-img" to handle disk image files from a wide range of virtualization tools and of course QEMU itself. The terminal command used to convert a raw image file using qemu-img would look similar to this depending on the expected outcome: `qemu-img convert -O qcow2 nameOfImageFileToConvert.img nameOfConvertedImageFile.qcow2`.

This produces a converted image file in the most recent QEMU disk container format. It can also be given a parameter that turns on compression to reduce the file space taken. For converting to formats compatible with VMware and VirtualBox, `qemu-img convert -O vmdk nameOfImageFileToConvert.img nameOfConvertedImageFile.vmdk` produces disk images usable by the VMware virtualization tool suite or by Virtual Box. Conversion directly to the native format of Virtual Box and Virtual PC is also possible, however in tests run in the laboratory VirtualBox often failed when emulating older operating systems and Virtual PC is mostly deprecated. For all experiments involving the alteration of the hard disk image file it is a good idea to also keep the original. Some experiments such as starting QEMU directly on a container file to check if the installed operating system boots, may render the image unusable for further experiments.

After producing the proper container format for the particular virtual machine or emulator the next step is to configure the virtual or emulated machine to boot from the image. The steps involved in this process depend heavily on the tool in use: While QEMU is normally configured solely via a command line most of the tools require a setup procedure using some sort of Graphical User Interface (GUI) and these are available for QEMU also). The virtual hardware configuration should match to the capabilities of the original operating system and it's supported hardware components. For example it is easier to adapt a Windows 95 based system to the emulated PCnet network adapter than the Intel e1000 network adapter as the PCnet adapter is of the same era as Windows 95 and has driver support included with the operating system.

4.3 Test Booting of Images

Once the emulator or virtual machine is properly configured, and the disk image file correctly linked to it, an attempt can then be made to boot the original system in the emulator or virtualization software. If the virtual environment boots successfully with no errors then pending further tests the process has been successful and is now complete. There is a possibility that the environment may simply fail to boot from the beginning. This can often be caused due to changed disk geometries or different BIOS capabilities of the original and the virtual machines. When this occurs the boot sector may need to be re-enabled.

The system can often be made bootable by using a valid operating system boot medium such as a Windows boot floppy disk or, for newer systems, the optical installation medium (e.g. the Windows CD-ROM). Another option when these issues are encountered is to create a system boot disk on the original system that was imaged. This has the advantage of matching exactly with the installed software. Alternatively disk images of boot-media for most operating systems can be download from the Internet. In the experiments that

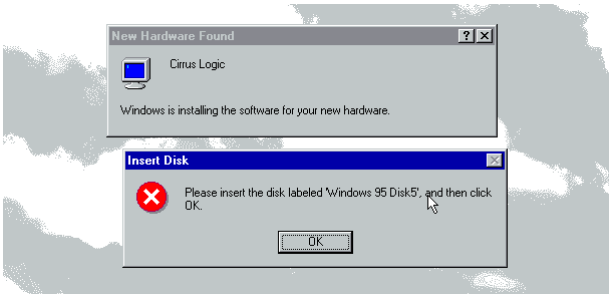

Figure 5: Reinstallation of drivers on emulated/virtualized hardware

were conducted, almost all of the system images were able to be booted directly within a QEMU-emulated machine and the other virtual machines tested. Only the Windows 98 system required a boot floppy disk to be used to restore the boot loader. This was achieved by loading the image with the boot disk in the emulated floppy drive, and typing: `a:` followed by `sys c:`. This reinstalled the necessary portion of the operating system core files and made the image bootable again. These actions might differ for other operating systems and could be more complex for newer systems like Windows XP, OS/2 or Linux. The next issue that may arise is an incompatibility between the installed hardware drivers and the new hardware provided by the emulation or virtualization software.

4.4 Finalize the Migration

Typically the emulator or virtual machine being used will get to a point at which it produces errors about missing hardware, or wrong drivers or will not boot into the original system's GUI at all. In order to rectify this, the next step required is to re-run the operating system's hardware setup procedure to re-detect the hardware configuration (Windows 95 and above). This usually triggers the reinstallation of drivers (Fig. 5) which in turn should give improved VGA, sound output and network access. As the drivers are often not part of original installation these need to be provided either by preinstalling them on the disk image or by making them available via the virtual/emulated CD or floppy drives.

Other systems such as Windows 3.X can require the Windows setup procedure to be run before loading the operating system in order to swap the video driver back to a basic VGA driver (Fig. 7, otherwise it may be impossible to load the OS in a meaningful way). If a higher resolution and/or color depth is desired this can then be achieved through changing the settings via the GUI once VGA mode has been enabled (Fig. 6). Unfortunately, successfully making the image bootable does not always mean the operating system will be able to completely load from the image. Hardware compatibility errors will often arise from attempting to run an operating system configured for one set of hardware on a different set of hardware. Typically the emulated machine being used will get to a point at which it produce errors about missing hardware, wrong drivers or will not boot into the original system's Graphical User Interface (GUI) at all. The solution to this (for Windows 95 and above) is to run the operating system's hardware setup procedure to re-detect the hardware configuration. This usually triggers the re-installation of drivers which in turn should give improved VGA, sound output and network access. As the drivers are often not part of original installation these need to be provided either by preinstalling them on the disk image or by making them available via the virtual or emulated CD or floppy drives.

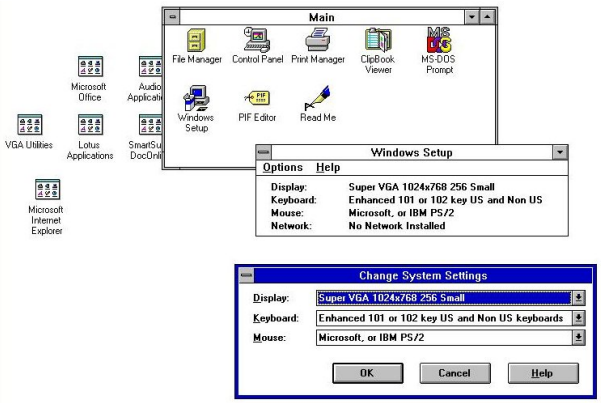

Figure 6: Alterations to Operating System Configuration

For newer proprietary operating systems, the original installation media or some equivalent is usually require to provide the standard driver set. Additionally, if the hardware changes the license might needed to reactivated. This requires the license key to be available to be re-entered. In order to make this step easier it is often possible to find software tools to read license keys read from the hard drive image from where they are stored on the disk, for example from the Windows system registry.

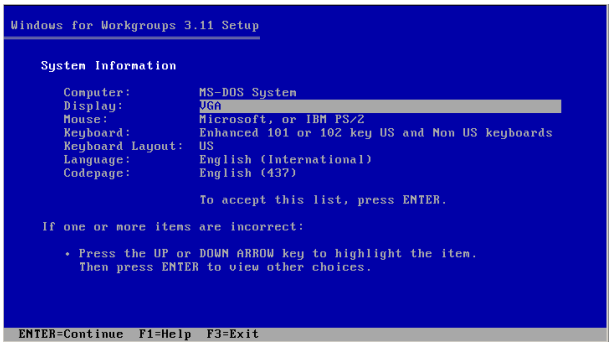

Figure 7: Altering Display Drivers in Windows 3.11

Fortunately, there is not a huge degree of variance between most hardware components in the different emulated hardware environments. Because of this, it should be possible to create additional disk image containers for each emulator or virtualization tool containing all of the relevant files that are needed to revive an arbitrary imaged original environment. Typical files which should be included are the installation files, relevant drivers and additional utilities which might help later users of the imaged systems. In most cases these files could be copied to a blank formatted hard disk image and such a hard disk could be attached as a further device to the virtual/emulated machine which would be automatically recognized by most, if not all, operating systems. At this point the environment should be ready for use.

5. POTENTIAL FOR AUTOMATION

The research conducted in order to inform the creation of the workflow outlined in this paper also enabled the testing of the general feasibility of dumping complete machines to be run in emulated hardware environments. One result of these investigations was the realisation that some of the tasks may be seen to be somewhat complicated for a non-technically trained practitioner (such as the use of the command line). Another finding was that there are many steps in the workflow outlined above that have the potential to be fully or partially automated, often by rather simple batch scripts run within an average Linux environment. Future research would be required to establish the best techniques for handling a wider range of systems and for dealing with more variable hardware. It would be beneficial to allocate resources to investigate how to automate some of the steps within the original environments. Here it might worthwhile to investigate whether the results of previous research into handling of interactive environments could be applied for automating some of these steps.

Step 1(.2) (Non-Intrusive) Disk Imaging could be automated through the creation of customized Linux distributions that are setup solely for the purpose of imaging old media. Linux distributions designed for digital forensics workflows [5] could

be modified for this purpose. This part of the process could be highly automated to a point where it would simply be a matter of inserting the disk or CD into the target computer, connecting a network cable and typing a few short commands, or making a few selections, to adjust the settings for the particular situation.

Step 2 Identify Hardware Requirements could be automated through the creation of an application to scan the disk image to identify most or all of the hardware requirements of the software environment installed within it. For example Windows operating systems often store information about the hardware that the operating system is installed on within the system registry files and these files could be queried to provide information on the appropriate emulation environment. If the machine is accessed via Linux for dumping, than listing of hardware components by using standard hardware detection tools like `hwinfo` or `lspci` is pretty straight forward.

Step 3 Select Emulation or Virtualization Software will always be a decision point, however intelligence could be added to make the decision as simple as possible by having the results of step 3 automatically compared to a database of virtual and emulated hardware provided by the different applications. Extended tool registries like [10] or [11] should be able to support this in the future.

Step 4 Modify Disk Image Format for Emulation or Virtualization Software could also be automated through the creation of an application that took the results of step 3 and automatically converted the disk image to the appropriate format. As this is run on a today's machine it is just a line to be added to a script running on the emulator hosting computer.

Step 5 Setup Emulated or Virtualized Hardware could be partially automated by mapping the results of the hardware identification in step 2 to the settings in the virtualization or emulation software being used. There will still likely need to be some decisions made about the configuration for each different environment but where generic enough this may be able to be fully automated.

Step 6 Test Boot of Image in Emulated or Virtualized Hardware has the potential for automation through a number of mechanisms. The error logs of the virtualization or emulation software could be analyzed to check for any boot-problems. It may also be possible to identify drivers by analyzing the disk image file to ascertain hardware conflicts in the installed operating system after an initial boot test, or by applying image analysis using the I/O interface (such as VNC or RDP) of the emulation or virtualization software to check for errors being presented in the virtual or emulated environment.

Step 7 Add Drivers could be further automated by the creation of custom applications for each operating system and virtual/emulated hardware combination that could be run on the virtualized/emulated environment to install the necessary applications. Media containing the necessary drivers could be attached to the virtual/emulated system and the relevant options could be automatically selected when the operating system asks for the necessary driver files.

Step 8.1 Re-Enable Boot Sector could be automated using the I/O interface of the emulation or virtualization software. This could be accomplished by the use of methods similar to those identified in previous papers published by one of the authors on using emulation for bulk migration [13]. This method uses an program running outside of the emulated or virtualized environment to automatically interact with the environment and select the relevant options when necessary during the boot-sector re-enabling process.

6. OTHER CONSIDERATIONS

Besides the technical procedures a number of additional issues should be considered by memory institutions dealing with emulation of original environments.

Technical Expertise. As in other domains of non-digital and digital preservation, specific expert knowledge is required to execute the workflow outlined above. This includes a basic understanding of computer architectures like x86 or the different Apple Macintosh platforms. Skills for disassembling old desktop or rack mount computers or laptops of various kind are needed to handle the hardware and in particular the hard disks with the required care. In order to successfully execute this workflow, future digital archivists and archaeologists will need to have a good knowledge of the operating systems of the past, at least to the degree that they are able to identify the vital parts to make them executable again on the emulated or virtualized hardware. The authors were able to handle the aforementioned DOS and Windows environments. Nevertheless dealing with OS/2 and BeOS (two other operating systems popular on the x86 platform mid to end of the 1990s) was placed out of scope of the tests as specific expert knowledge was not available. While the x86 and Motorola CPU based Apple platforms were reasonably mainstream, the preservation of other architectures like Sun Solaris on Sparc or DEC Alpha would require experts both of the platform itself and of the emulators involved.

Legal Issues. Beside the technical challenges a range of legal implications need to be considered. While in theory the moving of a software installation from one computer to another without duplicating it was not prohibited by the original license terms of many older software applications and operating systems, the whole domain is fairly undefined as yet [12]. Newer licensing terms are more restrictive and often require a re-licensing action when changing hardware. Only the license agreements of newer operating systems tend to explicitly deal with the option of virtualization. Furthermore, the preservation of entire original environments requires additional workflows beside the technical ones described in this paper. Not only does the hard disk need to be copied, but all the licenses of the installed software components need to be transferred to the receiving memory institution. This shouldn't be a problem in most cases, as the software is typically deprecated and not used anymore by the donor or transferring agency. But it implies or requires additional procedures to cover such things as the transferring over of license material and software keys. Those items need to be stored with the metadata of the original environ-

ment in order to best ensure their long-term preservation. The possibility and implications of running several instances of the same machine need to be considered also as this, while potentially very useful, will likely be illegal in most jurisdictions without the purchase of additional license keys.

A completely different legal issue exists regarding the privacy concerns of the donors. There are unlikely to be many privacy issues for government archives taking transfers of official government equipment, as users are and were typically prohibited from using their machines for private matters. Unfortunately it is a completely different situation regarding computers donated by famous authors or politicians. As the system imaging uses well established procedures of computer forensics the contents of resulting image file is often complete in a way that is beyond the imagination of the original user [7, 5]. Depending on the file-system and applications originally used, a great deal of additional information can be included than might be expected by a donor. Files are not deleted instantly in many file-systems but blocks containing them simply marked as empty. They don't get overwritten until the file space is required. Thus many deleted or other types of temporary files can easily be restored from the disk image files by experts. Thus special routines might be required to "clean" the resulting file image.

Authenticity Challenges. Part of the intent of the procedure suggested in this paper is to ensure the ability to preserve access to versions of digital objects that can be verified to have maintained their significant properties and thus verified to be authentic. [3]. In general there is a greater likelihood that digital objects preserved using this method will have full information integrity due to the objects being presented to users using their original software environments. This outcome is challenged to a varying degree in three ways:

- The original system can be altered by the re-installation of necessary hardware drivers and required adaptations to the new virtual hardware environment.
- Privacy concerns might require the removal of sets of files or ultimately, the cleaning of certain file-system blocks.
- Legal restrictions may require the removal of once installed applications or software components.

These threats to the authenticity and integrity of the environments, and the objects that are viewed and interacted with using them, are important but are not devastating to the outcome. In most cases the changes that need to be made to the driver files, or system set-up files, will not cause any difference to be manifested in the final performance of the replicated environment. Furthermore, in the cases where the aim of the system replication is to produce a representative desktop environment from an organization, it can be argued that such environments ought only pass the same tests of compatibility with the new hardware as the original environments did. In such cases it may be sufficient to confirm that the operating system is adequately running on the new emulated hardware as that would have been the only test any particular PC had to pass in the organization from which the replicated environment was representing an example. Usually most of those adaptations do not affect the rendering experience of objects. Nevertheless lower or higher screen resolutions, different color depth or the availability of 3D rendering may alter the overall experience of certain object classes. One counter argument to this is provided by the fact that in most organisations that created older digital objects, any particular user was only expected to have a representative rendering environment for the objects that they dealt with. For example in a most organisations there would have been many different hardware environments being used to render the same digital objects through the use of sharing technologies such as floppy drives and/or network connections. For this reason it can be argued that preservation practitioners should not have to fulfil any greater requirements when undertaking the preservation of the objects created in such environments. In other words, it can be argued that preservation practitioners only need to provide a representative rendering environment as that is all that an average user from the time of the creation of the objects was expected to have.

For the other two challenges, the redaction procedure could be built in to transfer/donor agreements such that memory institutions had the donor or transferring agencies approve any data redaction or software redaction that had to take place. Furthermore challenge three may well end up not being an issue for some institutions if laws can be changed to enable emulation to take place without the burden of license payments.

7. CONCLUSION

By demonstrating the feasibility of x86 system imaging and reproducing the imaged information environments (Fig. 8) in emulated hardware environments, the authors demonstrated an alternative understanding of complex objects that required a specific type of preservation solution. The focus is shifted from the characterisation of objects in attempts to make them reproducible in completely different digital ecosystems, to the preservation of the whole original environment in which the objects were created, managed or viewed. Using this new technique no specific knowledge of the object and creating application is required. As emulation and virtualization of the x86 architecture is well established, the described method might be used to simplify certain preservation workflows. By utilizing this approach the current diverse methods for the handling of different types of digital object have the potential to be simplified into a standard procedure for preserving a whole computer.

Preserving significant properties and object experience using emulation is discussed in [2], [8] or [9]. Depending on the object it may be desirable to alter the emulated hardware configuration to its needs instead of adapting the original environment to the hardware set provided by the emulator. This would be achieved by writing additional components for emulators so that they emulate the specific hardware of the environment that has been imaged. This is definitely a more costly option: if the demand for emulation increases more funding will be sought and found to provide just such solutions. It is also true that there is potential to share the costs in any such endeavour so as to get great benefits for all involved for little relative cost to any particular contributor.

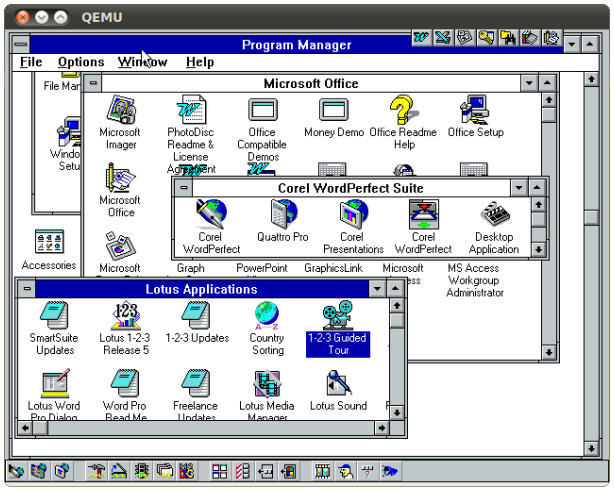

Figure 8: Imaged mid-1990ies Compaq desktop (Fig. 3) Windows 3.11 system running in QEMU

There is little difference between preserving a representative Windows 3.11 PC desktop from a government agency and preserving the system images of the portable computers of famous authors or important politicians. The workflow outlined in this paper can be applied to both situations.

The number of operating system and computer architecture combinations is much smaller than the number of file formats or complex digital objects like computer games to be considered. And, the emulation expertise could be easily shared between memory institutions. The workflow for replicating installed application and information environments onto emulated or virtualized hardware that is outlined in this paper should be able to be immediately tested and integrated into the digital preservation business processes of organizations where the necessary expertise and equipment already exist. For those where the workflow appears daunting or difficult it may be possible to obtain staff with the necessary expertise either temporarily or permanently if this is likely to be a regular process. For those where expertise is not likely to be available for some time, it should still be considered as an option when acquiring digital objects as there is significant potential to automate much of this workflow such that the expertise required will be minimal at a future date.

Automation of the aspects of the workflow that are identified above should be a primary research and development objective for the digital preservation community. The options and cost savings that the availability of replicated installed application and information environments provide, though not discussed in detail in this paper, are too large to be neglected. In spite of all the benefits of the approach outlined in this paper, a weak point still exists in the inability to be sure of the permanent availability of suitable emulators. While the number of short-term solutions in the virtualization sphere is quite high, a long-term, comprehensive digital preservation-aware emulator is still to be created. However in the meantime the number of good open source emulators could very well bridge the gap or grow into sustainable solutions [14].

8. REFERENCES

[1] Geoffrey Brown and Kam Woods. Born broken: Fonts and information loss in legacy digital documents. *International Journal of Digital Curation*, 6(1), 2011.

[2] Mark Guttenbrunner, Christoph Becker, and Andreas Rauber. Keeping the game alive: Evaluating strategies for the preservation of console video games. *International Journal of Digital Curation*, 5(1), 2010.

[3] Helen Hockx-Yu and Gareth Knight. Automation of flexible migration workflows. *International Journal of Digital Curation*, 3(1), 2008.

[4] Aaron Hsu and Geoffrey Brown. Dependency analysis of legacy digital materials to support emulation based preservation. *International Journal of Digital Curation*, 6(1), 2011.

[5] Matthew G. Kirschenbaum, Richard Ovenden, and Gabriela Redwine. *Digital Forensics and Born-Digital Content in Cultural Heritage Collections*. Council on Library and Information Resources, Washington, D.C., 2010.

[6] Mary J. Loftus. The author's desktop. *Emory Magazine*, 85(4):22–27, 2010.

[7] Sumit Paul-Choudhury. Digital legacy: Respecting the digital dead. *New Scientist Online*, 2011.

[8] Dan Pinchbeck, David Anderson, Janet Delve, Getaneh Alemu, Antonio Ciuffreda, and Andreas Lange. Emulation as a strategy for the preservation of games: the keep project. In *DiGRA 2009 – Breaking New Ground: Innovation in Games, Play, Practice and Theory*, 2009.

[9] Thomas Reichherzer and Geoffrey 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.

[10] The National Archives TNA. The technical registry pronom. Online, *http://www.nationalarchives.gov.uk/pronom*, 2010.

[11] UDFR Interim Governing Body. Unified Digital Format Registry – UDFR. Online, *http://www.udfr.org*, 2011.

[12] Jeffrey van der Hoeven, Sophie Sepetjan, and Marcus Dindorf. Legal aspects of emulation. In Andreas Rauber, Max Kaiser, Rebecca Guenther, and Panos Constantopoulos, editors, *7th International Conference on Preservation of Digital Objects (iPRES2010) September 19 - 24, 2010, Vienna, Austria*, volume 262, pages 113–120. Austrian Computer Society, 2010.

[13] Dirk von Suchodoletz. *Funktionale Langzeitarchivierung digitaler Objekte – Erfolgsbedingungen für den Einsatz von Emulationsstrategien*. Cuvillier Verlag Göttingen, 2009.

[14] Dirk von Suchodoletz, Klaus Rechert, and Achille Nana Tchayep. QEMU – A Crucial Building Block in Digital Preservation Strategies. In Wolfgang Müller and Frederic Pétrot, editors, *1st International QEMU Users' Forum – DATE 2011 Workshop*, Grenoble, France, 2011.