# A Practical Approach to System Preservation Workflows

Niklas Rehfeld, Ian Welch
School of Engineering and Computer Science
Victoria University of Wellington, New Zealand

Euan Cochrane
Independent Consultant
Melbourne, Australia

Dirk von Suchodoletz
Department of Computer Science
University of Freiburg, Germany

## ABSTRACT

Digital objects are often more complex than their common perception as individual files or small sets of files. Traditional methods of preserving these complex objects such as migration may not be suitable for maintaining access to them in an economically and technically feasible way. Many of today's preservation scenarios would benefit from a change in our understanding of digital objects. Instead of focusing on single digital files or small groups of files as they are commonly conceived of, computer systems in full should be considered. The preservation community could benefit from widening its collecting scope to include complex objects such at scientific desktops, databases, machines running networked business processes or computers of famous people such as authors or politicians. Such objects are not just interesting in their own right but also have to potential to provide a more immersive and contextually rich experience than simpler digital object.

In this paper we describe a workflow to be used for replicating installed application environments 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. We focus on the x86 architecture but we also discuss considerations for generalising the workflow to work with a wider range of architectures such as Macintoshes or Android smartphones.

## 1. MOTIVATION

Digital preservation has, until recently, mainly focused on preserving clearly identifiable, static objects like text documents, images or video streams. It has also focused on moving the content from the as-created files to new files that use different software to present the content to users, either immediately after a repository receives the files (normalization) or at some point in the future (migration). This strategy is turning out to be increasingly limited in the amount of digital artifacts covered and does not offer viable solutions to the preservation of dynamic objects such as databases, computer games, software and product development environments, scientific desktops or digital business processes. Standard preservation methods focusing on single objects can lose important contextual information or may render an object unusable because of missing components.

There are several compelling reasons for making images of entire information environments and maintaining the ability to interact with them over time:

**Electronic lab journals** Scientific desktops are the primary working environment for many scientists often serving as the electronic lab journal. More and more science funding agencies and journals require the preservation of the primary data of experiments or surveys to allow the future scrutiny of experiments or to enable researchers to build on data from older experiments [13, 4] and to supplement open data initiatives [1].

**Preservation of evidence** Most of today's products are designed on computers, and there are legal requirements for documentation on long-lasting industry products and the ability to prove certain facts about the products in the case of litigation. It can therefore be necessary to preserve design documents and tools for, for example, drugs in the health care industry or embedded systems in the automotive and aerospace industries [11]. These documents are often created using some form of software or product development environment specific to the industry. Preserving the development environments can add vital context to the design processes as well as ensuring that documents can be interacted with and toolchains re-run in the future.

**Personal archives** Famous persons' software environments from devices like desktop computers, mobile phones, laptops or tablets should be considered for preservation. Preserving such complex environments in turn enables the preservation of the whole context of the contained artifacts, letting researchers experience old information environments created by politicians, artists or other famous people.

**Databases and business processes** Databases are complex, dynamic and often interactive [7]. In addition, they are typically customized for particular purposes. They are usually not just a set of table descriptions and contained data, but are used in many larger applications such as Content Management Systems (CMSes)

or business processes. The business logic is often hidden in the database front-end or applications accessing the database. As such, databases often can not be easily migrated into a preservable format without risking the loss of significant information or without significant cost involved in verifying the preservation of the integrity of the content post migration. Business systems need to be preserved as part of companies' or organizations documentation processes and might need to be reproduced e.g. for liability cases.

Prototypical digital environments could be produced to provide permanent "viewers" for digital objects that can easily be maintained over the long term and are known to be compatible with the objects. For example, an institution might provide a viewer representing a typical desktop machine from a particular technological era. Such an environment would help to support a general preservation strategy capable of dealing with non-static digital objects and complete software and hardware environments. A system image would automatically contain all required software elements like fonts for documents or codecs for multimedia objects.

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. It can become increasingly attractive not to try to somehow "extract" the object from its environment but preserve the environment as a whole, especially for complex objects created or executed on one or more dedicated machines. 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 organizations 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 [5, 12] helps to understand the envisioned process.

Deploying full system preservation triggers a number of related questions, like how the digital artifacts should be properly described, what information needs to be captured at the point of ingest of such an object and how the results can be tested and quality assured. As the required procedures to perform a whole system ingest can be quite complex and demanding, the whole procedure should be as automated as possible and the archivists should be supported to the widest degree possible.

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 has the potential to be changed by following the steps described in this paper: They constitute 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, librarian or other less technically skilled digital preservation practitioner.

## 2. RELATED WORK

Digital objects are traditionally 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 file" 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 interact with or render the wider object [6].

Memory institutions have become increasingly aware of the challenges of the preservation of complex, dynamic and often interactive artifacts. The idea to preserve entire digital working environments is not new, but has gained relevance in several domains [8, 10]. One of the first applications of the approach of preserving a famous person's installed information environment was demonstrated most successfully by the team at the University of Emory's Manuscript Archives and Rare Book Library. The archivists have preserved an image of the hard disk from Salman Rushdie's early 1990s Macintosh desktop and currently use an emulator to access it. Such a setup gives the library user, whether researcher, fan or otherwise interested, a much richer experience of the authors working environment than just some files of finished work or never published fragments.

Archives NZ demonstrated in 2011 the feasibility of x86 system imaging and reproducing the imaged information environments of various machines dating from the early 1990s up to the mid 2000s in emulated hardware environments [16]. Using this approach, 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 is ready to be used for a number of different preservation tasks. Using 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.

The concepts presented in this paper rely on the existence of appropriate hardware emulators. Emulators are applications running on modern systems to reproduce original hardware solely in software [15]. Thus, emulation helps in becoming independent of future technological developments.

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 emulated environments, the workflow outlined here 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 they were even more sustainable and viable as permanent solutions. Virtualization products may be seen as being quite short-term solutions from a dig-

ital preservation perspective however they are practical in so much as they are available now and can be used to solve current problems.

Hardware emulation is not new to computer science. Emulators have existed for quite some time. Therefore, the list of developed emulators is astoundingly long and covers a fairly wide range of platforms. Especially popular computer architectures are well covered. For not yet supported platforms it is still possible to produce an emulator from scratch [5]. Prominent examples in the Open Source community are projects such as Basilisk II [1], SheepShaver [2], QEMU [3], MESS [4] or MAME [5]. Not every emulator, however, is suitable for the needs of a long-term digital archive. Requirements of the respective archiving organization need to be differentiated, e.g. a national archive will likely require different computer platforms to a computer games museum.

# 3. SYSTEM REPLICATION WORK FLOW

Computers of different architectures all still have similar characteristics as they conform to the von-Neumann or "stored-program computer" model. The permanent storage of a computer system usually contains all necessary software components to boot the machine into working state allowing interaction with humans or other computers. This permanent storage can include the hardware ROMs as well as the secondary storage such as hard drives or flash storage on microprocessors. While the information on ROMs such as the BIOS is sometimes difficult to extract and preserve, they are generally interchangeable and are usually included as part of the emulation software [6] Therefore the most important component to preserve is the information on the secondary storage devices. This would typically include the operating system as well as the user applications and data that is the primary target of the preservation effort.

Depending on the environments needing to be preserved, practitioners need to consider preserving both single machines and multiple networked machines. Single machine configurations are found for typical personal devices such as laptops, desktop computers, smartphones or tablets. Multi machine configurations can be expected when handling, for example, CMSs with database backends, business systems or networked applications. In those cases the minimum required set of machines to be preserved needs to be established. Before the archivist takes over or the user of the preservation target triggers the system replication process a couple of preparation steps are to be run.

## 3.1 General Preparation and Machine Identification

To ensure authenticity, integrity and privacy the donor should backup the system and then run some standard system administration and sanitizing tasks like removing unnecessary files and folders as well as deleting irrelevant user IDs. They should defragment the filesystem if possible, and ensure that the system is working properly. In the next step they put the machine into a preservation-ready state by, for example, adding required drivers for the target emulator.

For proper onward system reproduction the preserved system needs to be pre-identified. The general type and era of machine needs to be established both for the selection of the OS to run the preservation tools, and later on the selection of the emulator. The main things which need to be established before an OS or emulator can be selected are the processor architecture (e.g. if it's an Apple, this could be Motorola M68000, PowerPC or Intel x86 architecture) and the available peripherals (e.g. CD-ROM or floppy). Emulators will be chosen on the basis of their compatibility with the processor architecture and their device support for peripherals.

Once these hardware components are identified, the next step is to identify the operating system that resides on the image. Ideally this is done by the donor or the archivist before the system is archived. However, if this is not possible, there are existing (open source) tools that will help with this, such as `os-prober` [7] from the debian-installer team.

## 3.2 Deciding on the Target Emulator

Depending on the preservation target there may be more or fewer variants and versions available for the different platforms. While systems like the ZX Spectrum, Apple II, or Sparc Classic were well defined in their properties, other platforms like the x86 architecture provide a much higher variance. While the former allow for 1:1 matching to an emulator the latter ones do not necessarily have an emulator matching all hardware features exactly. With the operating system identified, this information, along with the hardware metadata gathered during the system imaging procedure, can be used to select and configure appropriate emulator hardware. As emulators typically only emulate a small set of hardware variants, it is unlikely for platforms like x86 that the emulated hardware will be the same as the original hardware. This means that some sort of mapping between the original hardware and the emulated hardware will need to take place. One approach to this is to categorize the hardware and operating systems by their historical era. This makes sense as a first approach, as it is most likely that an operating system will have support for a hardware component that was around at the same time that the operating system was current.

## 3.3 Source Identification

After selecting the relevant machine(s) for system preservation the relevant sources of the permanent storage need to be identified. Depending on the preservation target there exist two methods for system imaging: Intrusive or non-intrusive [16]. Non-intrusive imaging can be done in several ways, including in-system imaging, restoring from backups and using a special purpose OS to boot the preservation target machine. All methods depend on the preservation target in different ways. In-system imaging relies on the capabilities of the operating system and the filesystems in

[1]See `http://basilisk.cebix.net/`.
[2]See `http://sheepshaver.cebix.net/`.
[3]See `http://qemu.org`.
[4]See `http://mess.org`.
[5]See `http://mamedev.org/`.
[6]E.g. QEMU emulates SeaBIOS and Plex86 BIOSes for the x86 architecture or OpenBIOS for the Sun Microsystems architecture.
[7]See `http://joeyh.name/code/os-prober`.

use. Special filesystems of newer systems (e.g. XFS) allow taking consistent snapshots without interference from running processes. This method should be suitable for newer preservation targets with suitable tools installed for system imaging. Depending on the architecture, in-system dumps may fail because of limitations like restricted access to the complete storage, open file handles during the operation and cached blocks preventing a clean system image. An alternative is to retrieve a full system backup from a backup system, if the institution the machine was running in uses a suitable backup strategy. The most versatile and standardizable method is to boot a specially prepared operating system running on the preservation target hardware, which provides the imaging functionality. As the latter method can cover the majority of past systems it is described in this paper in more detail.

Intrusive imaging involves removing the target hard drive(s) from the original machine. This should be feasible for most laptops, server and desktop machines of various kinds, but might be impossible for highly integrated hardware using onboard or on-chip permanent storage. Nevertheless, it 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:
3. Connect the hard disk to an USB-IDE/SCSI/SATA adapter and plug this one into a machine running Linux that the image is to be "dumped" (written) to.
4. 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 the appropriate operating system used for the imaging.

This requires a special purpose machine providing all necessary hardware interfaces to connect the removed preservation target drives which is pretty straight forward for IDE and SATA. For SCSI a larger variety of connectors can be expected, but adaptors bridging various standards are available. As the machine which takes the drives to be imaged 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. The USB option 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 the latter option which may be important in some contexts. Both methods depend on the continued availability of adapters from current (or future) drive connectors to legacy drive connectors. This may involve memory institutions keeping a set of so-called "rosetta machines", which have both modern and legacy connectors on them, in order to transfer the disk images [9].

## 3.4 Special System Dump OS

Non-intrusive imaging requires the practitioner to boot the preservation target using a special purpose OS. This Dump Helper OS (DHO) enables the practitioner to make an image of the all relevant media containing the preservation target's operating system, application programs and data. In addition to imaging, the DHO provides standardized functionality to gather the various meta data on the contained files and the whole environment as well. For these reasons we primarily concentrate on this approach in the remainder of this section.

The DHO needs to conform to a number of requirements:

1. Supporting and booting on the relevant hardware of the preservation target(s)
2. Allowing readable access to all relevant permanent storage (e.g. hard or solid state disk drives, flash memory)
3. Block wise copying of the permanent storage over a network connection
4. Secure transfer to the archival system

The system should be easy to use for archivists and other persons involved and allow a high degree of automation.

The DHO will use the organizations' network to effect the transfer of the image and metadata to the archive system. By default, our focus is upon TCP/IP networks although future work might investigate the use of older or alternative transport protocols such as AppleTalk. Several alternatives exist for assigning a network address to the DHO instance. Where the preservation target is permitted to connect to the organizational network either a fixed IP address assigned by the system administrator or the Dynamic Host Configuration Protocol (DHCP) can be used to automatically assign the IP address. Alternatively, the preservation target may be connected directly to the archive system using something such as a cross-over network cable and IP address manually assigned to allow direct communication.

The DHO must either be of the same era as the preservation target or be backwards compatible to allow the tools to make use of the target's hardware in order to create images of attached storage and in order to transfer the images across the network. In addition, compatibility is required in order to support tools for collecting metadata related to hardware.

Linux is a suitable operating system for that purpose. It is Open Source, can be compiled for a wide range of hardware [8] and cooperates with a number of different boot loaders [9] required for the different architectures which are to be imaged.

# 4. FIRST STEP: ACTUAL MACHINE PRESERVATION

[8]Linux is among the operating systems supporting the largest number of different processors and hardware architectures.

[9]Different hardware architectures have different means to load the operating system kernel from different types of permanent storage.

## 4.1 Starting the imaging by booting the Dump Helper OS

The way that the information is transferred from the source machine will depend to some degree on the type of network that it is connected to. In general, the most likely protocol for transferring information is one based on SSH. This protocol allows a remote user to transfer files as well as control the machine they are connected to, which means that no work needs to be done on the physical source machine, other than booting it. The consequence of this is that the DHO can be made smaller as there is no need for a user interface on the source machine. Using SSH ensures the privacy and security of the transfer over any network. While in private networks or between directly linked machines a loose handling of SSH keys is possible.

For Internet connections, either the preservation organization could hold their own SSH keys and add them to the DHO when they create it, or the DHO could generate a random password on boot, which the archivist notes down and uses to access the source machine from the host (the one that will run the emulator...) machine [10]

## 4.2 Copying the Content

Once the permanent storage becomes available on a machine 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 for traditional hard or solid state disks the full command used is as follows: `dd if=/dev/sdx of=imagename.img`. Where `sdx` 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/sdx` command helps to identify the disk by its size

Different tools might be required to properly read from mobile devices embedded storage and deal with the fixed separation between ROMs implementing the operating system [11] and the user data. Tools to do this exist for some platforms. For example, the `nandroid` tool [12] allows both ROMs and user data to be backed up in a format that can be loaded into an Android emulator. However, by default only a superuser (`root`) is permitted direct access to the operating system and generally manufacturers do not provide the authentication credentials required to become a superuser. This has led to an arms race between manufacturers and the developers of applications for obtaining superuser access by exploiting security vulnerabilities in the phone's operating system. These applications are often both phone hardware and operating system specific and using them risks 'bricking' or making the phone unusable. In some cases, the best course of action might be to avoid potential damage by only copying the user data although there is a small risk that this data may not be accessible because it requires a custom application implemented by the manufacturer and stored in the phone's ROM.

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/sdx` 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 boot sector and partition table are need to be taken care of separately. Both images needs then to be merged appropriately by concatenating and editing the partition table from within the booted original system.

While the network transfer is properly authenticated and encrypted it does not employ explicit data stream integrity checking. To ensure identity of the source and destination binary stream both can be checksummed e.g. using MD5. 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.

## 4.3 Gathering Metadata

Any data that may or may not be retrievable from a disk image should be collected as metadata from the preservation target. It may be useful to collect as much information as possible about the original hardware of the machine in order to enable easy selection of the appropriate emulated hardware for the images to be preserved on. Depending on the age of the original machine, this can be collected with either `lshw` [13] on older computers or non-x86 machines such as PowerPC (pre SMBIOS), or for newer computers a more comprehensive output can be obtained using `dmidecode` [14] Other tools that would be useful to have are the standard linux system administration tools `lspci` and `lscpu`, which will give extra information on the attached PCI devices and CPU respectively. The most important devices that will need to be identified are the processor family, disk drives (in particular, the filesystem format such as FAT or EXT), and the graphics card, as these will be critical to getting the image to boot at all. Therefore a minimum requirement would be `lscpu`, `lspci`, and `fdisk`, as together these will provide all of the most important information.

Additionally to the technical metadata the running DHO can be used to generate file lists and fingerprints of each file for each file system or partition. This both helps to check and show that the contained files in the filesystem are exactly in the same state after the got imaged and re-run in the emulator. It proves that strategies like defragmentation, elimination of zero blocks or compression of the resulting image does not alter the contained filesystem.

[10] There is an additional security concern if the SSH server does not regenerate the host keys on boot, in that anyone with a copy of the DHO can masquerade as the 'real' DHO and launch a man-in-the-middle attack. Therefore it is recommended that either the host keys are regenerated at boot, or the DHO is kept secure, and the server key fingerprints are checked when the client logs in for the first time.

[11] The ROMs may have been customized by the phone manufacturer.

[12] See `http://code.google.com/p/android-roms/wiki/NANDROID\_Backup`

[13] See `http://ezix.org/project/wiki/HardwareLiSter`.

[14] see `http://www.nongnu.org/dmidecode/`

# 5. SECOND STEP: POST-IMAGING WORKFLOWS

After an image or set of images has been dumped from the original source or retrieved from a full system backup there are two ways to deal with the artifact further on. Either it gets attached to a virtual machine or emulator (after some preparation), and the original environment is run, and all further actions are done from within it, or the image is analyzed from the archivist's working environment externally, which means that the dumped system is not executed at that time. Methods and tools of digital forensics could then be applied to the disk images to gather a wide range of information such as (1) Block device structure and filesystem, (2) Contained operating systems and (kernel) versions, (3) Installed software components identified e.g. with the help of the National Software Reference Library (NIST) database [15] or (4) Hardware drivers which should get exchanged to run the image within an emulator.

Additional steps in disk image forensics could include run sanitary procedures, such as setting passwords to ones which get documented in the image metadata, the removal of unnecessary users and stripping the disk image of any private information if necessary. For example:

- To enable access to preserved system, there may be the need to reset or attempt to recover the passwords for the preservation target. Tools such as `chntpw` [16] for the Windows family of operating systems allow local passwords to be reset and can be used on an image without needing to load it into an emulator. Other tools, such as Medusa/Bruter/Metasploit smb_login again for Windows require a running system because they mount that attack over a network and so this must be done once the object has been emulated or hosted in a virtual machine.
- Removing unnecessary users to reduce space requirements or remove potential security risks to the preserved system by removing users with too much privilege or weak passwords.
- There may also be situations where we need to anonymize information on the disk, for example to remove sensitive references to phone numbers. Although this may be difficult to do without application knowledge there are forensics tools that can be used to perform a first pass. For example, `bulk_extractor` from the AFFLIB (Advanced Forensics Format Library) project allows strings to be replaced and requires only access to the raw disk image [3]. These an other tools are included in Bitcurator [17] which can be used to automate the workflow of carrying out standard tasks like detecting and wiping privacy related or other sensitive data.

With respect to post-imaging workflows, research questions remain to be answered with respect to the potential impact on the authenticity of imaged system. All procedures might pose a risk on the authenticity of the original image if it is impossible to prove that the clean-up did not alter relevant components in an unwanted way. For example, to show that changes to the file system at the block level does not change the actual content of the files themselves so they no longer are semantically the same as the target preservation system.

[15] See `http://www.nsrl.nist.gov/index.html`.
[16] See `http://pogostick.net/~pnh/ntpasswd/`
[17] See `http://www.bitcurator.net/`.

## 5.1 Image Processing and Authenticity

The image of the permanent storage of the preservation target is a logical block-wise copy and thus containing empty blocks or blocks containing deleted data. Thus, the digital artifact of interest is not necessarily the whole original system but, for example, just the database installed plus some additional tools, as was the case in the LINZ example [18] The installation might contain additional software, like an office suite or a couple of programs, the person using the machine found useful, but are of no further interest for future access.

The disks of the systems to be imaged are not necessarily of optimal size, that is, using 99 % of the available storage space. As the disk dumping procedure is agnostic of the content of the original disk and cannot distinguish between blocks containing meaningful or deleted data, it has to transfer the whole disk. In a worst case scenario the system resides on e.g. a 120 GByte disk consuming only a fraction of it leaving the rest empty. Thus, it makes sense to try to minimize the resulting image. In addition, there are economic reasons for this, as the secure long-term bit preservation of large images might pose a significant factor in costs. A size reduction could be achieved in a number of ways:

- Clean-up procedures triggered in preparation steps.
- Use digital forensic methods to get rid of blocks marked as unused but containing old data which is officially deleted by the means of the underlying filesystem.

In further steps the tools of virtual machines or emulators could be deployed to identify empty blocks and shrink the image to its minimal size. Another approach would be convert to compressed image types offered e.g. by VMware or QEMU. These procedures are methods which use identity transformations and therefore do not alter the filesystem layer of the preserved virtual disk. This would be provable by comparing the number of files and their fingerprints before and after the transformation. It would be desirable to have more abstract methods to prove that the significant properties of the artifact to be preserved are not harmed.

# 6. THIRD STEP: RE-RUN THE PRESERVED SYSTEM

After post-imaging procedures have been applied to the image, the image now can be made loadable into an emulator or virtual machine [19]. Here the emulated preservation target is the guest

[18] See e.g. the OPF blog post, `http://bit.ly/Qjcsac`
[19] An emulator allows execution of a target computer's programs using a mix of hardware and software features and does require any compatibility between the emulator's computer architecture because it interprets the program code whereas a virtual machine requires a compatible computer architecture so that it can execute a non-trivial subset of the target computer's programs directly on its processor [2].

| Year | OS | CPU | Other HW |
|---|---|---|---|
| 1978 | | 8086 | |
| 1981 | MS-DOS 1.0, PC-DOS | | ISA Bus, CGA Graphics |
| 1982 | QNX, NetWare 86, Xenix | | $3\frac{1}{2}''$ floppies |
| 1983 | | | 802.3 (10BASE5) |
| 1984 | Xenix 2.0 | 80286 | AT Bus, EGA Graphics |
| 1985 | Windows 1.0, Xenix 2.1/286 | CD-ROM | |
| 1986 | NetWare 286 | 80386 | PATA (IDE), SCSI, ATi GS |
| 1987 | Windows 2.0, Xenix 2.3/386 | OS/2 | VGA Graphics, FPM RAM<br>ATi Wonder, IBM 8514/A |
| 1988 | | | EISA Bus |
| 1989 | SCO UNIX R3, NetWare 3 | | SoundBlaster<br>ATi VGA Wonder |
| 1990 | Windows 3.0 | 80486 | Builtin FPUs, 10BASE-T, EDO RAM |
| 1991 | Linux 0.1 | | MIDI, ATi Mach8, S3 911 |
| 1992 | Windows 3.1, Plan9 | | VESA, ATi Mach32 |
| 1993 | NT 3.1, FreeBSD, NetWare 4 | Pentium | PCI |
| 1994 | Linux 1.0, NetBSD 1.0 | USB | |
| | | | ATi Mach64 |
| 1995 | Windows 95, OpenBSD | Pentium Pro | DVD-ROM<br>ATi Rage<br>S3 ViRGE<br>S3 Trio |
| 1996 | NT 4.0, Linux 2.0 | AMD K5<br>Pentium MMX | AGP |
| 1997 | | Pentium 2<br>Pentium 3<br>AMD K6 | AC'97<br>SDRAM<br>NVIDIA Riva |
| 1998 | Windows 98, NetWare 5 | Celeron<br>Intel i740 | PCI-X<br><br>S3 Savage3D |
| 1999 | Windows 98SE, Linux 2.2 | Athlon<br>Athlon XP | 802.11a<br>RAMBUS<br>NV GeForce256 |
| 2000 | Windows 2000, Windows ME, FreeBSD 4.0 | ]Pentium 4<br>TM Crusoe<br>TM Efficeon | DDR RAM<br>NV GeForce2<br>S3 Savage2000 |
| 2001 | Windows XP, Linux 2.4, OpenBSD 3.0, NetWare 6 | | NV GeForce3<br>ATi R100 |
| 2002 | | | NV GeForce4<br>ATi R200 |
| 2003 | Server 2003, Linux 2.6, FreeBSD 5.0, NetWare 6.5 | Pentium M<br>Athlon64<br>Opteron | SATA<br>DDR2<br>NV GeForce FX |
| 2004 | Windows XP SP2, Linux 2.10, NetBSD 2.0 | | PCIe |
| 2005 | Server 2003 SP1, FreeBSD 6.0, OS X 10.4<br>NetBSD 3.0, Novell OES 1.0 | | NV GeForce6<br>NV GeForce7<br>ATi R500 |
| 2006 | Server 2003 R2, Vista, OpenBSD 4.0 | Core<br>Core 2 | NV GeForce8 |
| 2007 | Home Server, Linux 2.20, NetBSD 4.0, OS X 10.5<br>Novell OES 2 | Phenom | ATi R600 |
| 2008 | Server 2008, FreeBSD 7.0<br>Vista SP1<br>Windows XP SP3 | Core i3/i5/i7<br>Atom<br>Phenom 2<br>Via Nano | DDR3<br>NV GeForce9<br>ATi R700 |
| 2009 | Windows 7, Linux 2.6.30, FreeBSD 8.0, OS X 10.6 | | NV GeForce 100 |

**Table 1: This table lists the main operating systems, along with CPUs and graphics cards of the time. Note that many operating systems are left out of this table, as they were not widely used. The CPU dates are the releases of the first computers with a that type of CPU in them. Linux versions are given as kernel version, as every distribution has a different versioning scheme. The 'other HW' column lists other notable hardware changes, such as new video card series, new architectures such as ISA/PCI/PCIX.**

and the machine running the emulator or virtual machine is the host. An emulator is the best choice for long-term preservation because it does not require the host's architecture to be the same as guest's architecture.

There are a large number of specialized emulators available for preservation targets for older machines because of their comparatively simpler architecture compared to the x86 architecture. For example, there are at least five different emulators [20] for the Digital Equipment Corporation's PDP-11 sold from 1970 into the 1990s. Some emulators such as those for the PDP include emulation of the original machine's firmware whereas others such as Kent's Emulated GS for the Apple IIGS platform [21] require copies of original ROMs containing the firmware.

For the x86 architecture, virtual machines (for example, VMware [22], VirtualBox [23]) and emulators (QEMU and JPC [24]) are available. QEMU is probably the most mature of the offerings and offers better performance than the Java-based JPC solution.

Emulators vary in terms of their support for emulated hardware. .For architectures with a large variance in hardware, such as x86 machines, the likelihood that an emulator will have the same hardware as the preservation target is very small. Therefore it might be necessary to install drivers for the emulated hardware, either before or after the image is booted. Each operating system has a different method of installing and registering hardware drivers. Fortunately, as the set of emulated hardware is small, there will be a manageable number of drivers for this hardware, even across a large number of operating systems. There are two general approaches to this, the first is to inject the drivers offline into the system image, the second is to install them off some form of external media attached to the emulator, once the preserved system has booted in the emulator.

The availability of the first option depends on the mechanisms used by the operating system to install drivers, as it may not be possible to inject drivers offline on all operating systems. For example in Windows systems this will involve copying the driver files into the system directory and modifying the registry.

The second option depends on the system booting on the emulated hardware without the drivers, at least to a state where the additional drivers can be installed. This is quite likely to be achievable, as most emulation software provides some "baseline" hardware for critical devices, which will run without extra drivers on many operating systems, e.g. generic, VESA-compliant graphics hardware.

Generally, first option is only needed when the critical devices such as graphics adapters or input devices are not supported at all by the operating system, so the system is essentially unusable. The second option is useful for optional devices such as sound cards and network cards, without which the system will boot, but may not be fully functional.

In either case the archiving organization will need to compile a collection of drivers for the emulated hardware that they are using, for each type of operating system that they will be archiving [14]. As mentioned above, typically the set of hardware provided by an emulator is quite small, so this is achievable.

To give a sense of the scale of the problem. Consider the QEMU emulator [25] which emulates the following devices (selectable from the command line):

- IDE-Controller supporting up to 4 drives (the drives are disk images on the host computer)
- IDE CDROM device (in the form of a CD ISO image, or a real CDROM device)
- Floppy disk controller supporting up to 2 drives (floppy disk images)
- Graphics card (either a Cirrus Logic GD5446 PCI, or VGA-VESA)
- PS/2 Mouse
- Ethernet network card (Realtek RTL8139 PCI or NE2000 PCI)
- A serial port (COM 1)
- A parallel port (LPT 1)
- Soundcard (Soundblaster 16 and/or ES1370)
- A USB-UHCI host controller (the Intel SB82371)

Table 1 shows the diversity of hardware for the x86, this means that when using QEMU we must either match to the closest emulated device present or inject driver support.

# 7. OPEN QUESTIONS AND CONCLUSION

The approach outlined in this paper is both effective and presents a solution with applications in a number of different contexts. However it also raises a few new questions: What exactly comprises the object which is to be preserved authentically? Are the numerous operations to reduce the original disk image size identity transformations? Could the preservation of the integrity of the preservation target's content be proven in an automated way? How much change of the original image on the block level is acceptable?

One perceived disadvantage of system imaging is the ratio of object to system image size. In cases where the outlined approach may be used it's value will have to be assessed against the cost of preserving such large objects (the disk images). From experimental experience it could be expected that in many cases the cost to understand, document, and migrate the database, along with the costs in providing meaningful access to the migrated databases without the custom GUI, may render the emulation or virtualization approach quite attractive in comparison. As a mitigation strategy to the object size challenge it could be tried to isolate the object to be rendered in a standard original environment, an approach pursued by the bwFLA project [26]. Such a standard environment might be used for many different objects of the same type and reduce the per-unit cost of storing any "rendered object" to a minimal value. That is a significant assumption for complex databases but should be straight forward for the likes of "stand-alone" MS-Access style databases.

The disk imaging components and tools identified, present a reasonably straight forward procedure. For a productive setting, however, user-support for system preservation tasks to be carried out require framework integration and defined workflows. bwFLA is integrating system preservation in a Digital Preservation (DP) framework, including workflows, tools and test-procedures.

If the artifact is clearly identifiable within a system image further strategies might help to gain significant size reductions. Especially, if there are a limited number of very similar but reasonably standard installations of e.g. databases. In those cases it would then be attractive to have a common base system. This could

[20] See (`http://www.pdp11.org`).
[21] See (`http://kegs.sourceforge.net`).
[22] See `http://www.vmware.com`.
[23] See `http://www.virtualbox.org`.
[24] The Pure Java x86 PC Emulator (See `http://jpc.sourceforge.net`.
[25] See `http://qemu.weilnetz.de/qemu-tech.html#Device-emulation`.
[26] bwFLA homepage `http://bw-fla.uni-freiburg.de/wordpress/?page_id=7`.

be achieved in a later step by running the original system in the emulator and use approaches such as monitoring the relevant objects to find out all components and files it uses. The gathered information could be used to finally reduce the image to a minimum set of files and applications or find out which components have to be present in a custom-made base system.

In the future system imaging workflows could be directly linked to standard backup procedures which could help to save additional steps in later preservation or allow a system recovery from the backup service without the need of the original machine.

The experiments showed that a certain "cooperation" of the original operating system is required for the objects to be properly preservable. Another issue might raise from the transfer of the license for the OS and the database engine. Both should be taken into consideration for objects which should be preservable in the future and could be made a requirement when implementing a project or putting out a tender.

Still, there are a couple of challenges to be solved to allow non-experts – even the IT professional of today is not necessarily familiar any more with the intricacies of driver installation in e.g. Windows operating systems or the boot loaders of OS/2 – to run these processes at manageable costs and efforts. Here, a cooperation between the various memory institutions could help to maintain the necessary knowledge and provide the resources to employ the necessary digital archivists.

# 8. REFERENCES


[1] Christine L. Borgman. The conundrum of sharing research data. *Journal of the American Society for Information Science and Technology*, pages 1–40, 2011.

[2] M. D. Canon, D. H. Fritz, J. H. Howard, T. D. Howell, M. F. Mitoma, and J. Rodriquez-Rosell. A virtual machine emulator for performance evaluation. *Commun. ACM*, 23(2):71–80, February 1980.

[3] S.L. Garfinkel. Automating disk forensic processing with sleuthkit, xml and python. In *Systematic Approaches to Digital Forensic Engineering, 2009. SADFE '09. Fourth International IEEE Workshop on*, pages 73 –84, may 2009.

[4] Pieter Van Gorp and Steffen Mazanek. Share: a web portal for creating and sharing executable research papers. *Procedia Computer Science*, 4(0):589–597, 2011. Proceedings of the International Conference on Computational Science, ICCS 2011.

[5] 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.

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

[7] IEEE. *Digital Preservation Challenges on Software Life Cycle*, 2012.

[8] Matthew G. Kirschenbaum, Erika L. Farr, Kari M. Kraus, Naomi Nelson, Catherine Stollar Peters, Gabriela Redwine, and Doug Reside. Digital materiality: Preserving access to computers as complete environments. In *Proceedings of the Sixth International Conference on Preservation of Digital Objects, San Francisco*, 2009.

[9] 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.

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

[11] Rudolf Mayer, Andreas Rauber, Martin Alexander Neumann, John Thomson, and Goncalo Antunes. Preserving scientific processes from design to publication. In George Buchanan, Edie Rasmussen, and Fernando Loizides, editors, *Proceedings of the 15th International Conference on Theory and Practice of Digital Libraries (TPDL 2012), Cyprus*, 2012. Springer.

[12] 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.

[13] Andreas Rauber. It research challenges in digital preservation – evaluation of healthcare institutions for long-term preservation of electronic health records. In J. Bote, M. Termens, and G. Gelabert, editors, *ENTERprise Information Systems*, 2011. Springer.

[14] Maurice van den Dobbelsteen, Dirk von Suchodoletz, and Klaus Rechert. Software archives as a vital base for digital preservation strategies. Online, *http://hdl.handle.net/10760/14732*, July 2010.

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

[16] Dirk von Suchodoletz and Euan Cochrane. Replicating installed application and information environments onto emulated or virtualized hardware. In *Proceedings of the 8th International Conference on Preservation of Digital Objects (iPRES2011)*, pages 148–157, 2011.