Modular emulation as a long-term preservation strategy for digital objects

Jeffrey van der Hoeven, Hilde van Wijngaarden

Koninklijke Bibliotheek, the National Library of the Netherlands
The Hague, The Netherlands
Jeffrey.vanderhoeven@kb.nl, Hilde.vanwijngaarden@kb.nl

Abstract. The use of emulation as a preservation strategy is often looked upon with skepticism. Although it may be the only way to render digital objects authentically in the future, emulation is thought to be too technically challenging and therefore too expensive and time-consuming. This line of thought has thus far prevented emulation from being developed for preservation purposes. However, the National Library of the Netherlands (KB) and the *Nationaal Archief* of the Netherlands are of the opinion that emulation-based preservation can be worth-while and needs further development and testing. This paper presents the results of a preliminary phase of the emulation project conducted at the KB. It explains and evaluates existing emulators and proposes a new model for emulation called modular emulation. This model provides the basis for the development of a working prototype in the future.

1 Introduction

Digital preservation does not end with the careful storage of digital objects. In order to keep these objects accessible, a continuous effort towards the development of strategies for permanent access is required. The usability of digital objects is threatened by the rapid innovations in computer technology. New systems and software supersede each other faster every year, making existing technology obsolete. This is becoming a problem in everyday life, but is specifically visible in the cultural heritage sector because its main focus is long term preservation.

Permanent access strategies can roughly be divided into two groups: migration and emulation. Migration is aimed at the digital object itself, and aims to change the object in such a way that software and hardware developments will not affect its availability. By changing or updating the format of an object, it is possible to render these objects on current systems. Emulation does not focus on the digital object, but on the hard- and software environment in which the object is rendered. It aims at (re)creating an environment in which the digital object can be rendered in its authentic form. The choice for either strategy is determined by the demands of future users and the types of digital objects that need to be preserved.

Although it is impossible to define all requirements of a future user at the time of preservation, it remains necessary to create future user scenarios. When a future user wants to browse through an old website, the representation of the digital object plays an important role. This representation is defined by five aspects: content, structure, context, appearance and behaviour, as shown in figure 1.1. If a future user is only concerned with the content, migrating the object will certainly suffice. However, if he or she wishes to experience the object as it was in the past, the representation needs to be reconstructed as accurately as possible. This is only possible within the original environment, which can only be recreated using either original hard- and software or an emulator.

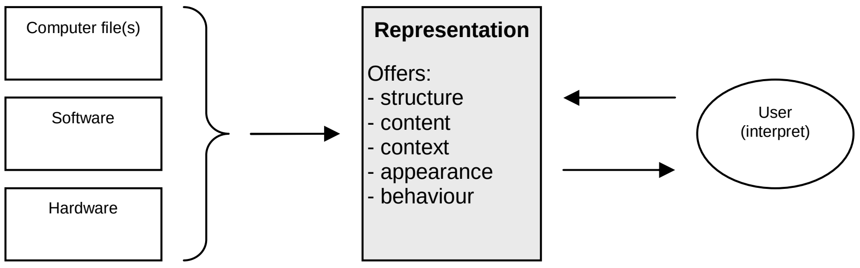

Figure 1.1: Representation model of a digital object

For relatively simple digital objects like text documents or images, migration (by means of format conversion or version upgrades) can work, although errors may occur. Migration is often denoted as a strategy for those objects with a short lifespan (up to ten years). The risk that subsequent migration cycles further decrease the authenticity of the object makes this strategy less suitable for the long term.

If migration is used to preserve fixed-format files or complex digital objects, like advanced text documents, spreadsheets with formulas or a computer application, it will lead to notable changes to the original object in the long term. As migration does not preserve the environment in which the objects are represented, functionalities like scrolling and searching through text, clicking on a cross-reference or running a query on a database, cannot be preserved authentically. Digital objects are increasingly dependent on functionality, so ignoring the environmental aspects disables the possibility of full reconstruction of the objects. A special concern are digital objects which depend strongly on the original environment. The only approach that can render them authentically is emulation.

1.1 The e-Depot

As National Library of the Netherlands, the Koninklijke Bibliotheek (KB) is responsible for the Dutch deposit library. The aim of the deposit library is to collect, catalogue and preserve all publications appearing in the Netherlands. This task includes publications on paper as well as in electronic form. In order to meet the challenge of electronic storage, the KB, in co-operation with IBM, has developed a large-scale storage system which meets the specific requirements of long-term preservation: the e-Depot [1]. In addition to long-term storage, permanent access strategies have to be developed in order to guarantee future accessibility of stored digital publications. That is why the KB has now established a programme to develop innovative tools and procedures to this effect.

The majority of electronic publications in the e-Depot are deposited and stored in Portable Document Format (PDF). For a long time, PDF documents were fixed-format text documents. However, newer versions of PDF can contain all kinds of embedded formats, in-text cross references, forms and even pieces of scripting code that increase the complexity of PDF [2]. In addition to PDF, other formats are accepted for storage in the e-Depot as well, for example Microsoft Office documents, TIFF images and HTML documents.

Another group of complex digital objects has been stored in the e-Depot since it became operational in 2003: interactive multimedia applications, which are supplied on CD- or DVD-ROM by publishers. These applications are installed on a standardised computer platform, which is called a Reference Work Station (RWS). After installation, a snapshot is taken from the hard disk's content by creating a disk-image of the installed application including the operating system. This image is stored in the e-Depot and can be reinstalled on the RWS at any time.

Although the RWS is only a temporary solution to rendering interactive multimedia, it provides researchers at the KB some time to develop a solution with a more permanent character. Moreover, rendering strategies also need to be developed for publications stored in PDF or other formats. The KB aims at keeping publications accessible in their original format because migration can alter the appearance of a publication leading to a possible change in meaning. Furthermore, migration does not offer a solution for more complex digital objects.

In order to achieve this goal, the KB conducts research into innovative permanent access solutions [3]. In co-operation with IBM [4], the KB has developed the Universal Virtual Computer (UVC) [5] to retain access to image formats of which the UVC specification and a demonstration tool is freely available at the IBM Alphaworks website [6]. Furthermore, the KB has recently started a project in co-operation with the Nationaal Archief of the Netherlands to build an emulator for digital preservation.

1.2 The emulation project

The choice for emulation as a preservation strategy is not undisputed, even though its possibilities are recognized [7]. Jeff Rothenberg, a strong advocate of emulation, states that emulation is cost effective and less error-prone than migration, because a single emulator retains access to many different digital objects, without the need to migrate each individual object periodically [8]. Some others are more sceptic and argue that emulation is too complex and therefore too expensive. Furthermore, intellectual property rights on hard- and software could restrict the development and usability of emulation [9]. Aside from these different views, one fact remains: emulation has never actually been developed and tested within an operational digital archiving environment. The KB and the Nationaal Archief of the Netherlands, apart from being convinced that emulation is the only option to preserve digital objects including full use of all functionality, also strongly believe that this strategy needs to be developed and tested in order to conduct a proper evaluation.

The first phase of the project, consisting of an exploratory study, has just been completed, setting the objectives for the next phase: building an emulator for digital preservation purposes. Research has been conducted to identify former and current projects in the field of emulation within and outside the digital preservation context. It offered better insight in different emulation techniques and existing implementations, and outlined the theoretical approaches for emulation-based preservation. As a result, the KB defined a new model for emulation in the field of digital preservation. Further on in this paper an overview of the results of this research is given.

2 What is emulation?

From 1999 until 2003 the CAMiLEON project (Creative Archiving at Michigan and Leeds: Emulating the Old on the New) [10] conducted research into the possibilities of emulation as digital preservation strategy. They defined emulation as [11]: “the re-creation on current hardware of the technical environment required to view and use digital objects from earlier times.” Emulation is the process of bringing digital objects back to life in their original environment on top of a different computer environment. This process is carried out by an emulator, which by definition of the Digital Preservation Testbed [12] “is a program that runs on one computer and thereby virtually recreates a different computer.” In this definition the word virtual denotes that the emulator functions like the original computer, but physically is not. The original computer is called the target (or client) platform; the computer that executes the emulator is called the host platform.

2.1 Emulation levels

Emulation can take place at three different levels: at application software level, at system software (operating system) level and at hardware level [13]. To emulate a particular level correctly, knowledge is required of its design and implementation. Due to the fact that application and system software are complex and often proprietary, emulation on one of these levels is difficult. Furthermore, emulating each software application separately requires many specific emulators.

Emulation can also take place by mimicking the hardware architecture, called *full emulation* (not to be confused with the term *hardware emulation*, which is a term that is either used for emulating hardware or emulation using hardware). Here, computer hardware, for instance a processor, is emulated by a surrogate in software or hardware. Although full emulation using hardware is possible, it is less flexible, costly and eventually would result in the same problems as we are facing today with obsolete hardware devices. In effect, the emulator in software is placed on top of the host operating system (OS) instead of running directly on hardware. The services and flexibility offered by the host OS, like the Application Programming Interface (API) [14], remain available. In this case, the emulator is a software program that runs on a host platform (hardware and OS) and recreates the hardware of the target platform under emulation, as depicted in figure 2.1.

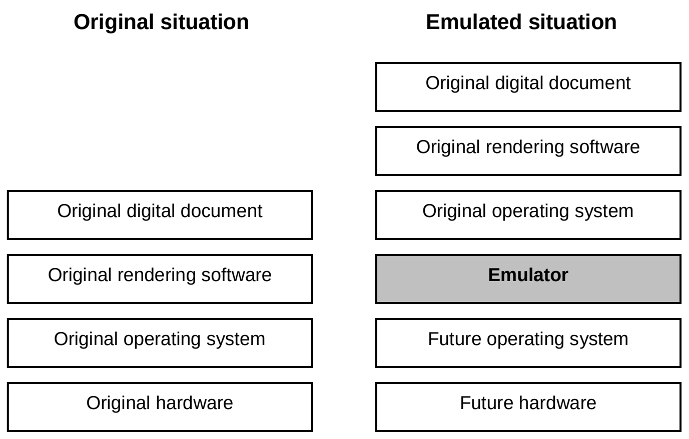
Original situation
Emulated situation
Original digital document
Original rendering software
Original operating system
Original hardware
Original digital document
Original rendering software
Original operating system
Emulator
Future operating system
Future hardware

Figure 2.1: Software-emulation-of-hardware

Although full emulation can be quite complicated, it has a more straight forward behaviour than emulating higher levels. For example, higher level features such as graphical user interfaces and multi threading (running multiple applications side by side) are difficult to emulate accurately. Emulation of a hardware platform does not incorporate these aspects, but requires the reproduction of the functional behaviour of the original hardware in such a way that the original software is not able to distinguish between emulation and reality. Because hardware specifications are well defined and are usually available, this behaviour is easier to reproduce than that of an OS or software application. Moreover, this approach retains the original OS, applications, drivers and configuration, which secures authenticity of the original environment.

Henceforth, emulation is defined as emulation of hardware by means of software, running on top of an operating system that itself runs on a hardware platform.

2.2 Practices

Although emulation has hardly ever been used for digital preservation purposes, it is certainly not a new technique. It has been applied for many years and serves many purposes. Numerous emulators have been developed and can be found on the Internet [15]. The available emulators can be categorized according to their purpose:

- Game computer platforms
- Historic computing
- Business efficiency
- Cross platform emulation

Game computer platforms

The nostalgic aspect of computer games has led to initiatives to keep older game console platforms alive by means of emulation. Key aspect in emulating game consoles is the game play, in other words the realness of the emulated environment.

Examples of platforms	: Atari, Nintendo, Sega and many other consoles.
Examples of emulators	: MAME, Meka

Historic computing

The same nostalgic aspect can be found by running old computer platforms. Like emulators for game consoles, several emulators exist for historic computer platforms. Moreover, emulation is sometimes the last hope for recovery of long forgotten information stored in some obscure format on obsolete media. Emulators make these media accessible again by running the original operating system and driver support.

Examples of platforms	: Commodore, MSX, PDP.
Examples of emulators	: SIMH [16], blueMSX

Business efficiency

Emulation can also lead to more efficiency in business environments. An emulator can create a virtual environment in which new applications can be tested and debugged safely. Virtualization, which differs slightly from emulation, is more restricted to the underlying hardware architecture, but allows a user to run multiple operating systems on the same hardware without significant loss of performance. Several corporate emulation and virtualization software applications are available.

Examples of platforms	: x86, PPC and SPARC
Examples of OS	: Windows, Linux, MacOS, Solaris
Examples of emulators	: VMware, MS Virtual PC, SoftMac, SVISTA

Cross platform emulation

Emulators are also built to host different platforms on other platforms, i.e. cross platform emulation. Especially in the open source community where Linux compatibility towards other operating systems is desirable, numerous initiatives exist.

Examples of platforms	: x86, PPC and SPARC
Examples of OS	: Windows, Linux, MacOS, BSD, BEOS, Reactor, DOS
Examples of emulators	: Bochs [17], QEMU [18], PearPC [19]

3 Digital preservation approaches

An emulator for digital preservation needs to meet specific demands. The central issue is how to ensure that an emulator developed today, can still run in the distant future. The emulators that have been developed for the purposes as described earlier, are dependent on the host platform and cannot simply be used for preservation purposes. Other important aspects concern the accuracy of an emulated platform, such as timing and performance: to be sure an emulator can render digital objects in an authentic way, it has to be tested while the original platform still works.
If we look at emulation in the digital preservation context, three main approaches can be described.

3.1 Stacked emulation

With platform-dependent emulation, a specific hardware platform is emulated directly on top of a current platform and OS. The advantage of applying this kind of emulation is the efficiency that can be gained by specifically developing the emulator for one particular host platform. This generally enhances performance and functional behaviour of the emulated platform. However, it lacks compatibility with other platforms and is less durable because the emulator is bound to one particular configuration of the platform. To keep these emulators running on subsequent platforms over time, stacked (or layered) emulation should be applied: the platform an emulator is built for in its turn needs to be emulated when it becomes obsolete (Figure 3.1) [11].

Stacked emulation generally results in a loss of performance, but other drawbacks exist as well. Each emulator strongly depends on its underlying environment, which means that functionality that is not available on the host platform and possible intermediate emulators cannot be offered on the target platform. Moreover, if one of the in-between emulators is corrupted or missing, the chain relying on this emulator is inevitably lost, resulting in inaccessible applications and documents.

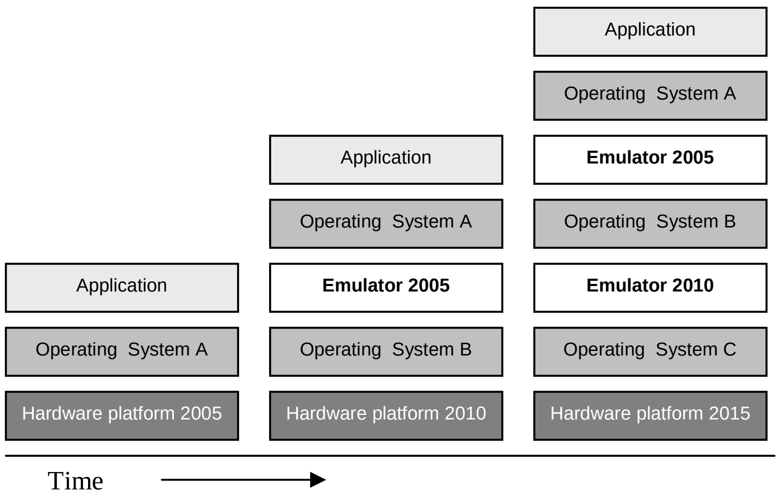

Figure 3.1: Stacked emulation over time

3.2 Migrated emulation

With migrated emulation the emulator is adapted to a different environment [11]. First, an emulator is created for one particular host environment. Then, if this environment changes, for instance the OS becomes obsolete, the emulator is translated to run on a new host environment. This translation process requires a compiler that is able to convert the source code written in language X into a binary executable that will work on the new platform. The original emulator is periodically migrated to subsequent environments, as shown in figure 3.2.

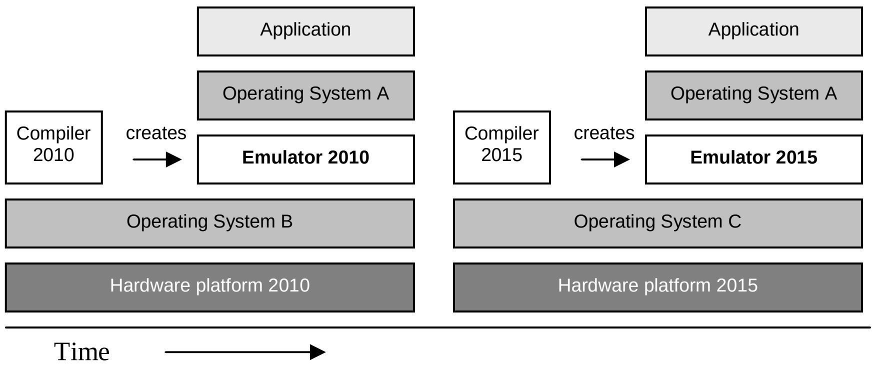

Figure 3.2: Migrated emulation over time

However, the source code of the emulator is not compiled today, but in the distant future. This can raise environmental problems because it is not guaranteed that a compiler for today’s favourite source language X is still available. And even if it is, the compiler's library may not be compatible with the original one, which can lead to malfunctions.

The CAMiLEON project took an attempt to build an emulator using a subset of the programming language C, which they called C-- [20]. The idea of this approach is that the C-- set only contains basic C functions that can be easily maintained in the future. Although CAMiLEON showed that this approach worked for emulating the old ICL 1900 platform, they also admit that it is inevitable that restrictions of C-- make certain aspects impossible. Since an emulator is an advanced application that relies heavily on communication with the host and target platform, migrated emulation is a strategy with high risks.

3.3 Emulation Virtual Machine

Ideally, an emulator is independent from time and platform. Jeff Rothenberg defined an approach to reach this independency by introducing an additional layer between the host platform and emulator, called an Emulation Virtual Machine (EVM) [21]. Together with an emulator specification and an emulation interpreter, emulators can be created to run on the EVM. According to Rothenberg, the EVM is stable over time and able to run on various host platforms. The emulator is no longer bound to any particular platform. Furthermore, it is possible to run multiple emulators on the same EVM. However, the side effect is that the EVM must be maintained over time and will be quite complex.

A variation on the use of a virtual machine is the Universal Virtual Computer (UVC), invented by Raymond Lorie from IBM [22]. Together with IBM, the KB worked out this concept and developed an operational UVC for image formats [23]. The UVC is a software program that recreates a simple general purpose computer and can easily be implemented on any computer platform now and in the future. Programs that are written in UVC language can be executed on the UVC in a platform independent manner. In the KB/IBM project, UVC programs (decoders) have been developed to decipher the original object and turn it into an XML-like structure. Next, this structure is used to recreate an original view of the object. This approach works well for image formats [5] and can be applied to all other digital objects which do not contain behavioural aspects, but it is not suitable for running advanced applications with user interaction and functional behaviour. To use the UVC for emulation, some major issues will have to be resolved. Apart from performance issues, an emulator has to be written in UVC language and the UVC needs to be adjusted to address more I/O functionality [24].

4 Conceptual model: modular emulation

Based on the evaluation of current emulators and the different approaches to applying emulation in a preservation context, a new approach for the realization of full emulation (emulation of hardware) has been developed. This approach will make it possible to render digital objects while retaining their full functionality over the long term. At this moment, it is a conceptual model that will evolve into a defined scenario during the KB/NA emulation project. The principle of this model is partly based on ideas of Jeff Rothenberg and Raymond Lorie. The use of a Virtual Machine (VM) and emulator specification are based on the ideas of Rothenberg, whereas the universal platform and component library are inspired by the UVC and its library of decoders by Lorie.

4.1 The concept

The model, as depicted in figure 4.1, shows a modular approach of emulation and is called *modular emulation model*. Modular emulation can be defined as:

> "Emulation of a hardware environment by emulating the components of the hardware architecture as individual emulators and interconnecting them in order to create a full emulation process. In this, each distinct module is a small emulator that reproduces the functional behaviour of its related hardware component, forming part of the total emulation process."

This modular approach stays close to the basic architecture of hardware, known as the Von Neumann architecture. It offers great flexibility in configuring new emulators by reusing modules as its individual hardware components.

The conceptual model consists of the following parts:

- Universal Virtual Machine
- Modular emulator
- Component Library
- Controller
- Emulator specification document

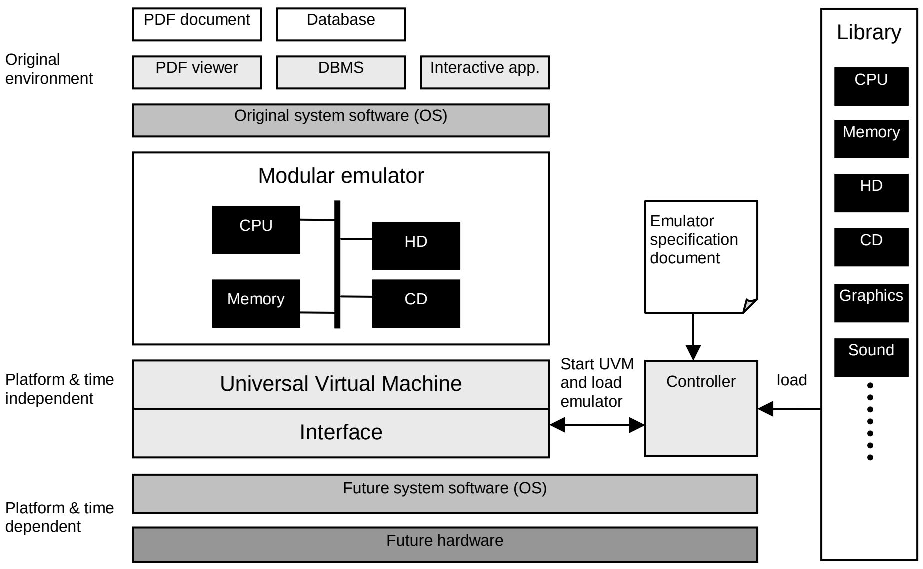
PDF document
Database
Original environment
PDF viewer
DBMS
Interactive app.
Original system software (OS)
Modular emulator
CPU
HD
Memory
CD
Emulator specification document
Library
CPU
Memory
HD
CD
Graphics
Sound
Platform & time independent
Universal Virtual Machine
Interface
Start UVM and load emulator
Controller
load
Platform & time dependent
Future system software (OS)
Future hardware

Figure 4.1: Conceptual model of modular emulation

Universal Virtual Machine

The basis of the modular emulation model is founded on the Universal Virtual Machine (UVM). The UVM is a platform- and time-independent layer on top of the underlying future host platform. As a result, programs that are developed for the UVM will continue to work even if the host platform changes. The UVM can be seen as an advanced version of the UVC. It not only consists of a general purpose processor and memory, but also offers additional functionality for input and output (I/O) communication with peripheral devices (like keyboard, storage, graphics, networking) between host and target platform. This will be provided by the interface layer.

Modular emulator

To execute the original system and application software on the UVM, an emulator is needed. The task of the emulator is to recreate the original hardware platform in software in such a way that the target software can run on it as it did on the original platform. Because of the versatility of hardware, system software and applications, the emulator needs to be flexible in its recreation of a particular environment. This can be achieved by designing the emulator in a modular structure. Each module emulates a specific hardware component in software on the UVM. A recreation of the target platform can then be realized by combining all necessary modules. Changing the target platform can simply be done by changing the configuration of the modules. Moreover, new platforms can easily be created by reusing existing modules in a different configuration. As technology moves forward, new modules can be created based on their analogy with actual hardware components.

Component Library
Building a modular emulator requires several modules. Therefore a library is needed, which contains all modules that can be used to build an emulator. The library is the owner and manager of all modules. It keeps track of the available modules and organizes it by type of components, such as: CPU, memory, storage, graphics, etc. In a second phase, the type of components can be further refined by refining the implementations. For instance, an initial implementation of a processor-module can be made compatible for a general x86 platform, whereas in a second phase this implementation can be refined resulting in two new modules, one for Intel Pentium and another for AMD Athlon.

The level of detail at which a module is implemented is left up to the programmer. However, each module aims to recreate the functional behaviour of the actual hardware component as good as possible. All modules are written in UVM language so they can run on the UVM platform. The library offers a minimum set of modules to create at least one target platform, but can be expanded with more modules over time.

Controller
The controller acts as the manager of all parts. It starts the UVM on the host platform and builds a bridge between the I/O handling of the host platform and UVM. The specifics are left up to the future programmer, because they depend on host platform characteristics. Subsequently, the controller starts the emulator and loads the required modules and settings.

Emulator specification document
With the library of modules, controller, UVM and modular emulator in place, the emulator can be told which hardware platform to emulate. This requires an emulator specification document, which describes the target platform by its components and significant properties. Based on this document, the controller loads the required modules into the modular emulator and sets their properties, like the speed of the central processor and the amount of memory required.
Figure 4.2 defines a simple emulator specification document. A definition of each component is given together with some significant properties. This document can also be structured using XML, as shown in figure 4.3.

CPU

• Chipset	: x86 compatible
• Clockspeed	: 700 Mhz

RAM

• Capacity	: 128 MB
• Speed	: 133 Mhz

Graphics adapter

• Bus type	: AGP
• Capacity	: 32 MB
• Max. resolution	: 1024 x 768
• Color depth	: 32 bit
• VESA support	: yes
• 3D rendering support	: no

Storage devices

Number of magnetics	: 1
Number of opticals	: 1

Magnetic

• Type	: hard disk
• Bus type	: IDE
• Controller interface	: ATA0 master
• Capacity	: 20 GB
• RPM	: 7200

Optical

• Type	: CD-ROM
• Bus type	: IDE
• Controller interface	: ATA1 master
• Read speed	: 24x
• Write speed	: 8x

Figure 4.2: A simplified human readable emulator specification document

```
<EMULATOR>
     <ARCHITECTURE>
            <NAME>Von Neumann</NAME>
            <TYPE>x86</TYPE>
            <COMPONENT>
                   <TYPE>CPU</TYPE>
                   <CHIP>x86</CHIP>
                   <CLOCKSPEED>700</CLOCKSPEED>
            </COMPONENT>
            <COMPONENT>
                   <TYPE>RAM</TYPE>
                   <CAPACITY>128</CAPACITY>
                   <SPEED>133</SPEED>
            </COMPONENT>
            …
     </ARCHITECTURE>
</EMULATOR>
```

Figure 4.3: emulator specification document in XML

5 Conclusions and future work

In the digital preservation context, there have not been any significant developments in the field of emulation since the proof-of-concept by Jeff Rothenberg and the work on the CAMiLEON project. So far, an emulator within an operational archiving environment has never been built, but much of the preliminary work has already been undertaken. Emulation has already been used for various purposes and has shown to be a viable technique to keep digital objects accessible. It is therefore possible to use (parts of) existing emulators to develop a new emulator that is time and platform independent. The modular emulation model offers the flexibility to create different target platforms while providing the host platform with a universal layer.

In the second phase of the emulation project, all significant properties that are part of the current rendering environment will be defined. These include aspects of functional behaviour, graphics and sound. Apart from this, a test set and test environment will be defined to compare the behaviour of original objects with the emulated results.

The modular emulator can be built incrementally. It is unnecessary to complete the emulator in total in order to be able to gain results: separate modules can be tested after a relatively short period. For instance, the module to emulate sound can be delayed until a later version, while other functionalities are tested first. The separate modules will consist of parts of existing emulators, available in the open source community. The next step will be to make the modular emulator platform independent by adjusting it to run on a virtual machine. The Java Virtual Machine will be used to this purpose until this virtual machine will be replaced by a more universal approach, i.e. a Universal Virtual Machine.

The possibility to share intermediate results with others, allow the KB and the Nationaal Archief to gather support for this strategy while also utilising new insights along the way. The KB and the Nationaal Archief of the Netherlands have recently started the emulation project and plan to develop a modular emulator during 2006 so a working prototype can be presented in the beginning of 2007.

Acknowledgements

We would like to thank Jeff Rothenberg (RAND Corporation), Jacqueline Slats (Nationaal Archief of the Netherlands) and Remco Verdegem (Nationaal Archief of the Netherlands) for taking part in the discussion that led to the design of the modular emulator.

References

1. Oltmans, E., van Wijngaarden, H.: Digital Preservation in practice: the e-Depot at the Koninklijke Bibliotheek. In: VINE, vol. 34 (1), (2004), 21-26
2. Ockerbloom, J.M.: Archiving and preserving PDF files. In: RLG Diginews, vol. 5 (1), (2001). Available at: http://www.rlg.org/legacy/preserv/diginews/diginews5-1.html (accessed 8 September 2005)
3. Digital Preservation Research, the National Library of the Netherlands. Available at: http://www.kb.nl/hrd/dd/dd-en.html (accessed 8 September 2005)
4. Van Wijngaarden, H., Oltmans, E.: Digital preservation and permanent access: the UVC for images. In: Proceedings of Imaging Science & Technology Archiving Conference, San Antonio, USA. Available at:

http://www.kb.nl/hrd/dd/dd_links_en_publicaties/publicaties/uvc-ist.pdf (accessed 8 September 2005)

5. Lorie, R.A.: The UVC: A Method for Preserving Digital Documents: Proof of Concept. The Hague, IBM and Koninklijke Bibliotheek (KB), 2002. Available at: http://www.kb.nl/hrd/dd/dd_onderzoek/reports/4-uvc.pdf (accessed 8 September 2005)
6. IBM, Alphaworks Emerging Technologies, Digital Asset Preservation Tool (2004). Available at: http://www.alphaworks.ibm.com/tech/uvc (accessed 8 September 2005)
7. Hunter, J, Choudhury, S.: Implementing Preservation Strategies for Complex Multimedia Objects. In: Proceedings at the European Conference on Research and Advanced Technology for Digital Libraries (ECDL), Trondheim, Norway, 2003.
8. Rothenberg, J.: Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation. Counsil on Library and Information Resources, Washington, USA, 1999. Available at: http://www.clir.org/pubs/reports/rothenberg/contents.html (accessed 8 September 2005)
9. Granger, S.: Emulation as a Digital Preservation Strategy. In: D-Lib Magazine, vol. 6 (10), (2000). Available at: http://www.dlib.org/dlib/october00/granger/10granger.html (accessed 8 September 2005)
10. CAMiLEON, Creative Archiving at Michigan & Leeds: Emulating the Old on the New. Available at: http://www.si.umich.edu/CAMILEON/ (accessed 8 September 2005)
11. Holdsworth, D., Wheatley, P.R.: Emulation, Preservation and Abstraction. In: RLG Diginews, vol. 5 (4), (2001). Available at: http://www.rlg.org/preserv/diginews/diginews5-4.html (accessed 8 September 2005)
12. Emulation: context and current status. Digital Preservation Testbed, The Hague, The Netherlands, 2003. Available at: http://www.digitaleduurzaamheid.nl/bibliotheek/docs/White_paper_emulation_UK.pdf (accessed 8 September 2005)
13. Rothenberg, J.: Using Emulation to Preserve Digital Documents. Koninklijke Bibliotheek, The Hague, The Netherlands, 2000. Available at: http://www.kb.nl/pr/publ/usingemulation.pdf (accessed 8 September 2005)
14. Halfhill, T.R.: Emulation: RISC's Secret Weapon. In: Byte.com, special reports, 1994. Available at: http://www.byte.com/art/9404/sec8/art3.htm (accessed on 8 September 2005)
15. Wikipedia: the Free Encyclopedia, List of Emulators (2005). Available at: http://en.wikipedia.org/wiki/List_of_emulators (accessed 8 September 2005)
16. Subnik, B.: Simulators: virtual machines of the past (and future). In: Queue vol. 2 (5), (2004), 52-58
17. Bochs, think inside the Bochs (2005). Available at: http://bochs.sourceforge.net (accessed 8 September 2005)
18. QEMU CPU Emulator (2005). Available at: http://fabrice.bellard.free.fr/qemu (accessed 8 September 2005)
19. PearPC – PowerPC Architecture Emulator (2005). Available at: http://pearpc.sourceforge.net (accessed 8 September 2005)
20. Holdsworth, D.: C-ing ahead for digital longevity. CAMiLEON project, University of Leeds, 2001. Available at: http://cedarsgw.leeds.ac.uk/CAMiLEON/dh/cingahd.html (accessed 8 September 2005)
21. Rothenberg, J.: An Experiment in Using Emulation to Preserve Digital Publications. Koninklijke Bibliotheek, The Hague, The Netherlands, 2000. Available at: http://www.kb.nl/coop/nedlib/results/emulationpreservationreport.pdf (accessed 8 September 2005)
22. Lorie, R.A.: Long-term archiving of digital information. IBM Research report, IBM Almaden Research Center, San Jose, Almaden, 2000.

23. Van der Hoeven, J.R., Van Diessen, R.J., Van der Meer, K.: Development of a Universal Virtual Computer (UVC) for long-term preservation of digital objects. In: Journal of Information Science, vol. 31 (3), (2005), 196-208
24. Lorie, R.A., Van Diessen, R.J.:Long-Term Preservation of Complex Processes. In: Proceedings at the IS&T Archiving Conference, Washington, USA, 2005