Emulation: To Be or Not To Be

Remco Verdegem; Nationaal Archief of the Netherlands; The Hague, The Netherlands
Jeffrey van der Hoeven; Koninklijke Bibliotheek; The Hague, The Netherlands

Abstract

Emulation is often looked upon with skepticism. Although it may be the only way to render complex digital objects in the future without affecting their authenticity and integrity, emulation is considered 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 digital preservation purposes. The Nationaal Archief of the Netherlands and the National Library of the Netherlands (KB) believe that a preservation approach based on emulation is worthwhile and that it needs further development and testing in order to establish its true value. This paper presents an overview of emulation as a preservation approach, a conceptual model of modular emulation, experiment results and goals of the emulation project.

Introduction

Digital objects are fragile. The debate for the best means of preserving digital objects over the long term has been under way for many years and will no doubt continue for years to come. Various theoretical solutions have been proposed, and research is conducted around the world to identify ways in which digital records can be authentically maintained whilst remaining accessible and usable over the long term. This paper focuses on the project of the Koninklijke Bibliotheek (National Library of the Netherlands) and the Nationaal Archief of the Netherlands to develop a preservation strategy based on emulation.

Digital Preservation

Digital Preservation is concerned with ensuring that digital objects that are created electronically using today's computer systems and applications, will remain available, usable, reliable and authentic in ten, to more than one hundred years, time. The core of the problem of digital preservation is that hardware and software, which were used to create and interpret the object, will become obsolete in the near future. Digital preservation consists of preserving more than just the object's bit stream. We must also be able to interpret the bit stream in order for the digital object to survive. Without interpretation, the bit stream is nothing more than a meaningless series of 0's and 1's. During preservation, the object's characteristics as defined by its context, content, structure, appearance, and behavior must be taken into account. Behavior is an aspect that is specific to digital objects and therefore requires the necessary attention when it comes down to authentically preserving the object over the long term.

There are a wide range of digital formats available and, to make preservation more complicated, different digital objects have different preservation requirements. These not only depend on their technical aspects, but also on the reason the object is being preserved, how long it needs to be preserved, the context and history of the object. The Nationaal Archief of the Netherlands has to execute the Dutch archival regulation [1], which stipulates what needs to be preserved depends on the requirements of the business process to which the record belongs. The Koninklijke Bibliotheek has a similar interest, although not by legislation. It is responsible for the cultural heritage of the Netherlands, which includes an increasing amount of digital material. In 2003 the e-Depot [2], the electronic repository of the KB, became operational, which at present contains over five million electronic publications.

Digital Preservation has different meanings for each digital object. Whilst it is often thought that digital preservation means preserving the object so that it is identical to its original representation, this is not always required. Not every aspect of a digital object has to be preserved – this depends on the business process in which the object played a role. In all cases however, the object must be preserved in such a way that it retains its integrity and its authenticity. This presents interesting challenges.

Digital objects are dependent on their environment. They rely upon the hardware and software that was originally intended to interpret them. When the environment becomes obsolete, perhaps within the space of a few years, the problem of how to read that object without its original hardware and software arises. It is unlikely that different versions of a software application will interpret the object in the same way, and this may well result in a change in the interpreted object (the visible or available view of the computer file(s)) that affects its (archival) integrity. Some data may be lost altogether; in other areas, data may be gained. It is not always possible or feasible to compare a new version with the original, so changes may go unnoticed. Any changes to the digital object may affect its authenticity and integrity, which in turn may affect its (archival and legal) status. Depending on the nature of the object and its use, this can cause problems, not least that of losing or misrepresenting history.

The main preservation strategies currently are: technology preservation; migration (backward compatibility; interoperability; conversion into standard formats); the use of XML, encapsulation; virtual machine software (e.g. UVC) and emulation. Emulation is the focus of this paper and is now discussed in more detail.

Emulation as a preservation strategy

The theory behind emulation is that the only way to ensure the authenticity and integrity of the digital object over the long term is to continue to provide access to it in its original environment, i.e., its original hardware, operating system and software application. Emulation does not focus on the digital object, but on the environment in which the object is created and rendered. This is achieved by recreating an environment in which the digital object can be rendered in its authentic form.

Using the definition of the Digital Preservation Testbed [3], emulation is a program that runs on one computer and thereby virtually recreates a different computer. In this definition the word 'virtual' denotes that the computer functions like the original computer, but physically is not. The original computer is called the target platform; the computer that executes the emulator is called the host platform. The program that recreates the target platform on the host platform is called the emulator.

The term "simulation" is often confused with – and sometimes even used as a synonym for – emulation, but we distinguish between the two terms here by noting that simulation describes what some other thing would do or how it would act, whereas emulation actually does what that thing would do. For example, an airplane simulator does not actually fly. That is, simulation generally involves the use of a model to understand, predict or design the behavior of a system rather than the practical recreation of that system's capabilities. In contrast, emulation is generally used to create a surrogate for the system being emulated.

A similar misunderstanding exists between emulation and virtualization. Although both techniques are capable of recreating a virtual environment on a host computer, virtualization is more limited in the freedom of choice of its target platform. Virtualization uses so called instruction interpretation and is thereby dependent on the underlying host platform. If, for example, an x86 host platform is used for virtualization, the target environment should support this same x86 architecture. This limitation is not present in emulation, thus making emulation both more flexible and platform independent.

Outside the context of digital longevity, emulation has proved to be useful and reliable. In previous computer systems, both IBM and Apple used emulation to make their newer hardware compatible with older software [4][5]. During the nineties emulation became the key to run nostalgic computer games, which were no longer accessible without the original hardware. Nowadays, emulators like MS Virtual PC [6], QEMU [7] and Bochs [8], and virtualization techniques used in VMware [9] have become very advanced and complex, capable of running complete operating systems like Microsoft Windows, Apple's MacOS or Linux with a wide support of device peripherals.

Emulation levels

Emulation is stated as a technique to create a virtual environment on top of an existing environment. However, this creation can take place at three different levels: at application software level, at system software (operating system) level and at hardware level [3].

Emulation at the application software level consists of writing one application program to do what another application program does. In the preservation context, this is essentially the "viewer" approach, in which new programs are written in the future to render obsolete digital formats. Instead of writing a single emulator for a hardware platform, the viewer approach requires writing a new program (or adding a significant new piece to an existing viewer program) for every distinct digital format. Because many formats are proprietary, this entails reverse engineering each such digital format. Such an approach is taken by the Multivalent browser [10]. Although it is possible to use the browser for opening various kinds of digital documents in a platform independent way, authenticity is not taken into account. This results in slight differences between the Multivalent view and the authentic representation.

The idea behind emulation at system software level is to recreate the operating system (OS) that is used by rendering programs for various digital formats. This requires a significant amount of reverse engineering, but even so, the result is not a program that can run other programs, since this is not what system software does. It merely provides facilities (user interfaces, file systems, interprocess communication, networking, etc.) that are used by programs when they run, and it allows invoking programs to be run (e.g., by double-clicking on their icons). This means that a program still requires a particular hardware platform, besides the OS.

Emulation at hardware level takes place by mimicking the hardware architecture. Software is used to emulate computer hardware on which original rendering software can run. We refer to this as the "software-emulation-of-hardware" approach, not to be confused with emulation using hardware whereby the emulator itself is a piece of hardware. From a preservation perspective, this latter approach is a short term solution, because physical hardware is still part of the rendering process.

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, emulating one of these levels is difficult. Furthermore, emulation of each software application separately requires many specific emulators.

Although the "software-emulation-of-hardware" approach can be quite complicated, it has more straightforward behavior than emulating higher levels. Because hardware specifications are well defined and are usually available, this behavior is easier to reproduce than that of system or application software. Moreover, this approach retains the original OS, applications, drivers and configuration, which secures authenticity of the original software environment.

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

Preserving the emulator

The use of emulation in the field of digital preservation has been limited. However, over the years several research studies have been performed which concerns the issue of keeping the emulator itself accessible. That is: how can we get our emulator programs to run correctly on each generation of future computer?

In principle, three strategies to this problem have been proposed: chaining (also called stacked or layered emulation), rehosting (or migrated emulation) and the use of a so called Emulation Virtual Machine (EVM).

Chaining is an approach which allows each emulator of one computer to run indefinitely once it has been implemented on only one other (successor) computer. If the successor computer is emulated on its own successor computer, any previous emulator can be run under a chain of emulators, as illustrated in figure 1. Although this approach will probably work well initially, a new emulator needs to be created each time a new platform becomes current. Furthermore, the expanding chain of emulators will introduce an increasing risk of unsupported functionality and unstable behavior.

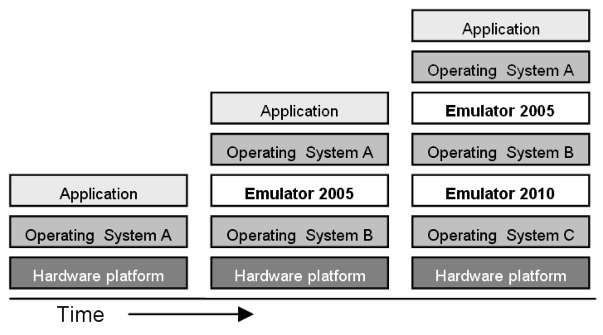

Figure 1. *Chaining*

As an alternative to chaining, emulators can be rehosted on successive future platforms. To do this, an emulator is written for a current computer platform. The source code of the emulator needs to be compiled (using a compiler), into an executable program for that platform. When this platform becomes obsolete, the emulator needs to be recompiled for a newer platform. Recompilation requires the original source code and a compiler that is able to operate on the new platform. This process is depicted in figure 2. The advantage is that the emulator is not dependent on previous emulators. On the other hand, it requires a compiler that should be capable of linking the required emulation functionality with the underlying host platform.

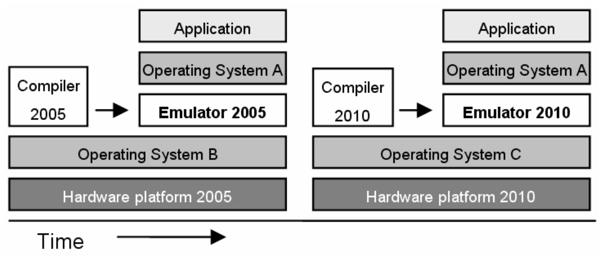

Figure 2. *Migrated emulation*

between 1999 and 2003 the CAMiLEON project (Creative Archiving at Michigan and Leeds: Emulating the Old and the New) [11] conducted research into migrated emulation and took into practice a subset of the programming language C, called C--. The idea of using this subset was that it would be fairly easy to maintain a compiler that is capable of translating C--code into platform dependent code in the future. Based on this knowledge they created an emulated version of the almost forgotten ICL1900 system. This implementation showed the potentiality of emulation, but still required functionality that was outside the scope of C--. Therefore it remains uncertain whether the emulator will remain accessible over time.

According to Jeff Rothenberg, a strong advocate of emulation-based preservation, this uncertainty can be avoided by using an "Emulation Virtual Machine" (EVM) [12]. This EVM will operate as intermediate layer between the host platform and the emulator. By offering a standard interface towards the emulator no matter which host platform is running underneath, the emulator becomes platform independent. In order to do this it is required that each emulator has to be written to run on the EVM. A very successful implementation of this approach is the Java Virtual Machine (JVM) [13], which makes Java programs portable across many different host platforms. A disadvantage of an extra layer is its complexity. Therefore, careful consideration is needed to decide what should be implemented in the EVM and what not.

Modular emulation

During 2004 and 2005 the Koninklijke Bibliotheek conducted preliminary research into emulation-based preservation. In cooperation with Jeff Rothenberg, a new emulation strategy has been developed called modular emulation [14]. The principles of this strategy are based on earlier ideas about the EVM of Jeff Rothenberg and the Universal Virtual Computer (UVC)-based preservation method of Raymond Lorie [15].

The modular emulation strategy stays close to the basic architecture of today's hardware, known as the Von Neumann architecture [16]. 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 behavior of its related hardware component, forming part of the total emulation process."

In figure 3 the conceptual model of this strategy is shown. Key features are its flexibility and platform independency, which will be outlined hereafter.

The modular emulation model consists of the following parts:

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

Universal Virtual Machine

The Universal Virtual Machine (UVM) is the base of the model. It is a platform- and time-independent layer on top of the underlying future host platform. This ensures that 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

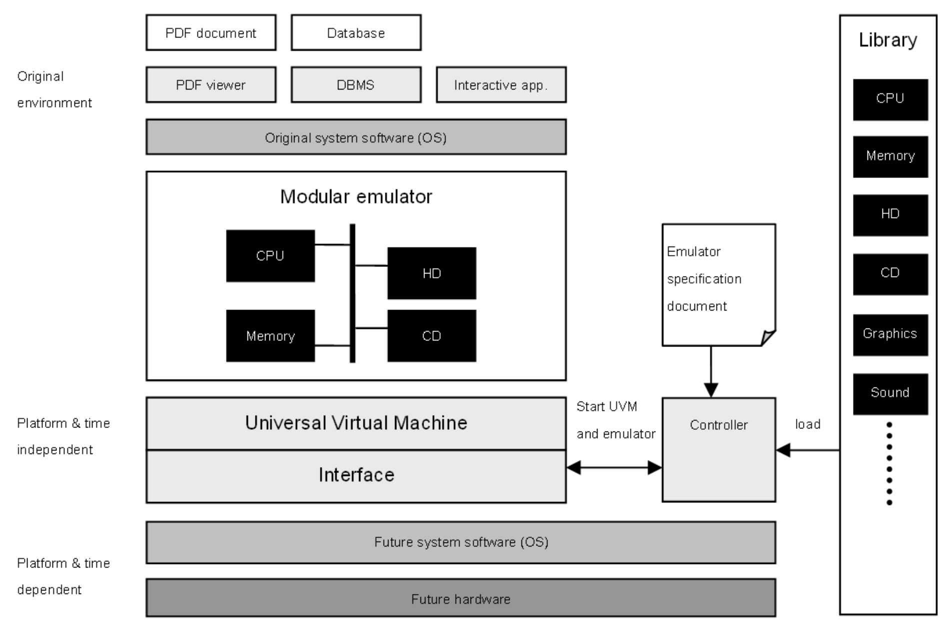

Figure 3. *Modular emulation model*

keyboard, storage, graphics, networking) between host and target platform. This will be provided by the interface layer.

Modular emulator

To execute the target system and application software on the UVM, an emulator is needed. The task of the emulator is to virtually recreate the hardware of the target platform in such a way that the target software can run on it as it did originally. 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. 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. 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. To keep track of the set of modules, a component library will be used. The library is the manager of all modules and will store and retrieve them with additional metadata. Organization can be done by type of component, such as: CPU, memory, storage, graphics, etc. It also should offer some kind of version control, so that modules can be enhanced over time.

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 faithfully as possible. All modules are written in UVM language so they can run on the UVM platform. The library should offer a minimum set of modules to create at least one target platform, but can be expanded with more modules over time.

Emulator Specification Document

To define which modules are required to create a modular emulator, a description of its configuration is needed. This will be stated in an Emulator Specification Document (ESD). It describes the target platform's components and significant properties.

Controller

Finally, the controller will interconnect all parts in the model. 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 interprets the ESD and starts the emulator, loaded with the required modules and settings.

Project modular emulation

The choice for emulation as a preservation strategy is not undisputed, even though its possibilities are recognized. Whatever the different views may be, one fact remains: emulation has never been actually developed and tested within an operational digital archiving environment. The Nationaal Archief of the Netherlands and the Koninklijke Bibliotheek are convinced that emulation is possibly the only option for future access to specific digital objects without affecting their authenticity and integrity. They agree that this strategy has to be developed and tested first, before its potential and limitations can be assessed.

As a result, the Nationaal Archief and the Koninklijke Bibliotheek have started a project to develop a preservation strategy that is based on emulation. The objective of this project is to develop an emulator that is modular and portable, following the conceptual model of modular emulation strategy. Modularity will help to minimize the effort required in emulating a new target machine, as it allows existing modules to be re-used or new modules to be added. Portability will be realized by using an intermediate layer, for which the Java Virtual Machine (JVM) has been chosen. Although the JVM itself is a complex layer, it is widely used and supported for a large number of different host platforms. Performance in Java is a point of concern, but with long-term preservation in mind portability is given higher priority than speed of execution, due to the fact that future computers will be much faster.

Emulation experiments

To explore the capabilities of existing emulators and virtualization software, a number of experiments have been performed using the Digital Preservation Testbed [17]. Four tools have been tested: VMware, MS Virtual PC, QEMU and Bochs. The test set consisted of three different types of digital objects: PDF documents, interactive multimedia publications on CD-ROM, and database systems. The test criteria were based on the five types of significant properties: structure, content, context, appearance and behavior. The results differed greatly. Whereas VMware is the only virtualization suite, it scored best overall on all aspects. Using emulators, hardly any difference was measured during experiments with PDF. However, the results with multimedia publications and databases varied a lot. MS Virtual PC performed well with multimedia, but disappointed during database experiments, whereas QEMU showed just the opposite. Bochs was less powerful in all fields, but was more reliable than QEMU.

Design & Development

Based on the test results a list of requirements has been defined. In cooperation with Tessella Support Services plc. and Jeff Rothenberg initial design has taken place. Development of the emulator will be done using prototyping. First, a core emulator will be developed, running a basic x86 instruction set and memory model. Subsequent steps will include interrupt handling, memory-mapping, display and sound, I/O-handling from keyboard and mouse, disk access, BIOS and implementing the boot sequence.

Besides the emulator, attention will be paid to the component library, controller and Emulator Specification Document (ESD). Depending on the available time, these parts will be implemented as well.

The possibility to share intermediate results with others, will allow the KB and the Nationaal Archief to gather support for this strategy while also utilizing new insights along the way. At the start of 2007 a working prototype of the modular emulator can be expected. The project will end April 2007.

References

[1] Regeling Geordende en toegankelijke staat archiefbescheiden (*Regulation on the Arrangement and Accessibility of Records)*, (February 2002). Available at: http://www.nationaalarchief.nl/images/3_2563.pdf

[2] e-Depot, Koninklijke Bibliotheek, The Hague, The Netherlands (2006). Available at: http://www.kb.nl/dnp/e-depot/e-depot-en.html (accessed 10 March 2006).

[3] Emulation: context and current status. Digital Preservation Testbed, The Hague, The Netherlands (2003).

[4] IBM 7000 series, Wikipedia.org (2006). Available at: http://en.wikipedia.org/wiki/IBM_700/7000_series (accessed 10 March 2006).

[5] J. Hoskins, Exploring the PowerPC Revolution! 2nd edition (1995).

[6] Microsoft Virtual PC 2004 (2006). Available at: http://www.microsoft.com/windows/virtualpc/ (accessed 10 March 2006).

[7] QEMU (2006). Available at: http://fabrice.bellard.free.fr/qemu/ (accessed 10 March 2006).

[8] Bochs (2006). Available at: http://bochs.sourceforge.net/ (accessed 10 March 2006).

[9] VMware (2006). Available at: http://www.vmware.com/ (accessed 10 March 2006).

[10] T.A. Phelps, P.B. Watry, A No-Compromises Architecture for Digital Document Preservation (ECDL2005, Vienna, Austria, 2005) pg. 266.

[11] CAMiLEON project (2006). Available at: http://www.si.umich.edu/CAMILEON/ (accessed 10 March 2006).

[12] J. Rothenberg, An experiment in using emulation to preserve digital publications, Koninklijke Bibliotheek, The Hague, The Netherlands , pg. 8. (2000).

[13] Java Virtual Machine (JVM, 2006). Available at: http://java.sun.com (accessed 10 March 2006).

[14] J.R. van der Hoeven, H.N. van Wijngaarden, Modular emulation as a long-term preservation strategy for digital objects (IWAW, Vienna, Austria, 2005). Available at: http://www.iwaw.net/05/papers/iwaw05-hoeven.pdf (accessed 10 March 2006).

[15] R.A. Lorie, Long-term archiving of digital information, IBM Research report, IBM Almaden Research Center, San Jose, Almaden (2000).

[16] Von Neumann computer architecture (2006). Available at: http://en.wikipedia.org/wiki/Von_Neumann_architecture (accessed 10 March 2006).

[17] Digital Preservation Testbed (2006). Available at: http://www.digitaleduurzaamheid.nl (accessed 10 March 2006).

Acknowledgements

We would like to thank Bram Lohman (Tessella Support Services plc.) for reviewing this article.

Author Biography

Remco Verdegem began his professional career in the area of information technology in 1989. In October 1998, he joined the Dutch State Archives' Service, where he was among other things responsible for the functional maintenance of the archival system for paper records. From October 2000 till July 2003 he was the project manager of the Digital Preservation Testbed project. In July 2003 the Nationaal Archief of the Netherlands adopted the Digital Preservation Testbed. Since April 2005 Remco is working as Senior Advisor Digital Longevity at the Nationaal Archief.

Jeffrey van der Hoeven started his work in the field of digital preservation during his graduation assignment on the Universal Virtual Computer (UVC) at IBM Netherlands N.V. in 2003. In 2004 he obtained his master degree in Computer Engineering at Delft University of Technology and started his career at the Digital Preservation Department of the Koninklijke Bibliotheek. He conducted research into emulation-based preservation and joined the emulation project in 2005.