Seminar
Long-term Preservation
Emulation: A Strategy for preserving Authenticity

Ingo Schwarz
1070438
06—08—2009

Responsible person:
Prof. Dr. Uwe M. Borghoff
Carer:
Dipl.-Inform. Nico Krebs

Institute for Software Technology
Department of Computer Science
Universität der Bundeswehr München

Contents

1 Introduction

Nowadays people seem to think that archiving information digitally is the best and most integrity-sustaining way to save data. It seems logical that handwritten papers degrade over time whereas for instance a word document can be easily duplicated and altered and the file does not decrease in quality. In Addition, the physical space needed to store such objects eventually becomes smaller as storage density increases. As a result it has become a natural course of action to use digital documents for preserving books, newspapers, scientific papers, government and corporate documents.

Now what is the problem? Suppose we use a computer, equipped with an operating system and all necessary software to create a random digital document. How can it be guaranteed that this setup will be available and kept working over long periods of time? Living in an age of rapid technological advance new platforms pop out of the ground like mushrooms and older platforms can become obsolete somewhat quickly. This means that one has to solve the problem of degradation of digital documents primarily caused by switching to newer computing platforms. There are several approaches to a solution of which emulation is to be illustrated in further detail in this paper.

After introducing some means for long-term preservation, the concept of emulation will be described in further detail by referring to various technical approaches. Subsequently, current projects on long-term preservation using emulation as a key strategy will be mentioned.

2 Means for long-term preservation

There are several methods at one's disposal for keeping digital documents which are running danger of becoming unreadable legible. A short summary of the most commonly used ones shall follow.

2.1 Migration

"Migration is the periodic transfer of digital materials from one hardware respectively software configuration to another or from one generation of computer technology to a subsequent generation. The purpose of migration is to preserve the integrity of digital objects and to retain the ability for clients to retrieve, display, and otherwise use them in the face of constantly changing technology."[1] Advantages and Disadvantages are described in further detail in papers by Robert Berger and Falko Krause which were created parallel to this seminar paper.

2.2 Emulation

Emulation is the replicating of functionality of an obsolete system. The essential idea behind emulation is to be able to access or run original data on a current platform by running software on that current platform that behaves like the original platform. Creating an emulator involves encapsulating certain pieces of information. A logical view of this encapsulation is shown below.

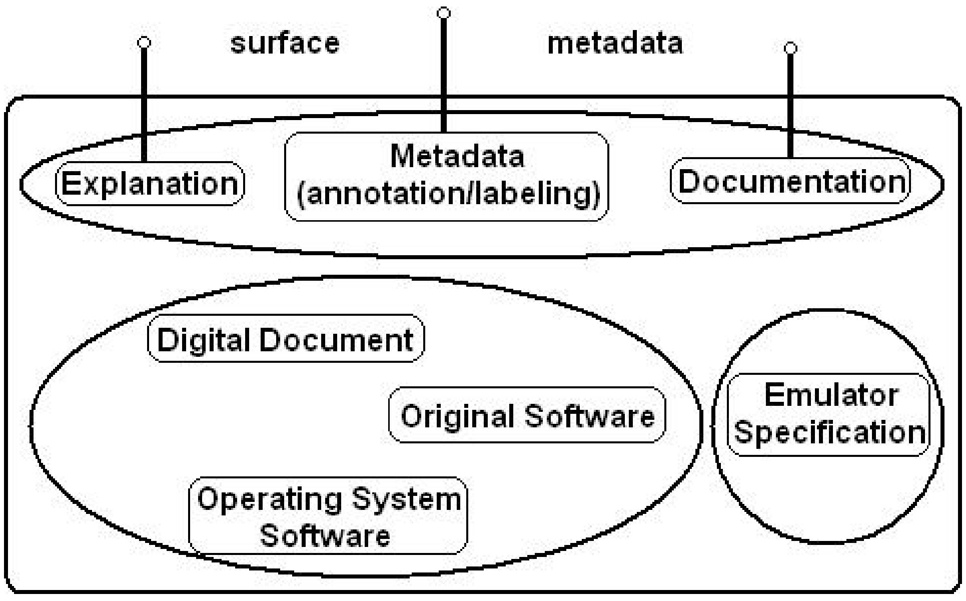

Figure 1: Logical view on encapsulation

The first kind of information to be encapsulated includes the document, its software environment and the required operating system software. Additionally, any other software that is needed to run that piece of software which is needed to open the encapsulated digital document has to be included. Technically, the bit stream(s) representing one or more original files is saved so that it can be accessed by its original software.

The second bundle of information compromises a specification of an emulator for the document's original computing platform. The herewith included difficulty is that the emulator cannot be an executable program, since it must be created without knowledge of the future computers in which it is supposed to run. In addition, a specification of all attributes of the original hardware platform is demanded in order to recreate the behavior of the original document when it is run under emulation.

The third type of information in the encapsulation of a document consists of explanatory material including labeling information, annotations, metadata of the document and its history and finally documentation for the software and emulated hardware that was encapsulated. From the preservation point of view, this last type is most crucial since this is the actual interface to a user. In case these descriptive components one day become non-human-readable, the emulator itself has faced the disaster of obsoleteness.[7]

There are a number of ways how to encapsulate information. Some of these are safer because more information is stored whilst others involve somewhat more of a risk since storing less information means there is a danger that this emulator is unable to function correctly.[7] Imagine some of the metadata and documentation to be missing. The emulator is then not preserved for a long time because at some point of time, the user will be unable to understand the emulator due to missing explanatory material.

Obviously, this practice makes it possible to maximize authenticity because the original object is never changed. Furthermore, only one emulator has to be created for one type of data object whereas in the migration approach it is necessary to convert every single object one after the other, thus increasing expenditures in terms of time and labor. However, emulation has certain practical boundaries when it comes to working with complex environments. Even more difficulties arise when system specifications of either side, old or new platform, are unknown.[7]

2.3 Universal virtual computer

The universal virtual computer (UVC) deals with the notion of using a virtual machine by the means that it creates a layer between the underlying computer platform and upper lying software. Then software, developed for running on top of the virtual machine, is then able to operate on a wide variety of computer systems for which a virtual machine exists. The attribute that makes the UVC universal is that it is general enough to continue to be relevant in the future. The key to achieving this is that the UVC must be kept as simple as possible.[6] The universal virtual computer solution will be described in further detail in the paper by Gerrit Kahn which was created parallel to this one.

3 Three technical approaches to emulation

Emulation can be done at three different levels: application software level, operating system level and hardware level.

3.1 Hardware emulation

The first emulator was actually a hardware emulator, developed by IBM in 1962. Back then the IBM System/360 was a new family of computers designed to cover a wider range of applications for both commercial and scientific purposes. This technological forward leap confronted all involved people, from computer architects to the purchaser, with the problem of old mainframe computers becoming obsolete which would have meant that all software had had to be rewritten for the new system. And because IBM's existing customers had a large investment in software that ran on now outdated machines, many new models offered the option of microcode emulation of the customer's previous computer.[11] As an example, Apple's "Rosetta" is to be mentioned which is a translation process that runs an "Apple PowerPC" binary on an Intel-based Macintosh computer. The PC's kernel detects whether an application has native binary and triggers Rosetta in case it does not.[1]

Nowadays it is frequently done to emulate hardware with other hardware to create simpler or lower-power versions of existing processors, though it can just as well be done to create faster or more powerful versions of popular or older machines. This is the technically lowest level on which one can emulate. The idea is to modify a new piece of hardware in a way that it is possible to run older programs with a probably totally different set of instructions. This so called microcode emulation either can be accomplished by using hardware description languages and then simulating the newly created hardware by software or by actually building it on semiconductor basis.[4] It is well known from numerous computer science lectures that any piece of hardware can be described by defined hardware description languages. This hardware specification is then stored as microcode and can be simulated on current computers. While an emulator runs an old machine on a new one and recreates its behavior completely, an emulator logically rebuilds the old machine probably leaving out some "unimportant" features. Besides the fact that simulation and emulation have become a standard way of functional verification, they seem to be of importance when it comes to

[1] http://developer.apple.com/documentation/MacOSX/Conceptual/universal_binary/universal_binary_exec_a/universal_binary_exec_a.html

long-term preservation since saving hardware means preserving it for the future.[5]

From the preservation perspective, so called “software-emulation-of-hardware” or simply “full emulation” is often employed. In this way, computer hardware is emulated by a software surrogate.[8]

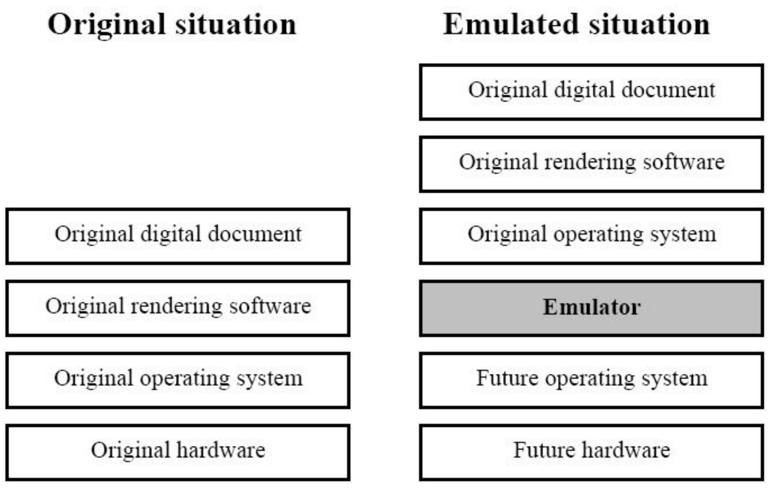
Original situation
Emulated situation
Original digital document
Original rendering software
Original operating system
Original hardware
Original digital document
Original rendering software
Original operating system
Emulator
Future operating system
Future hardware

Figure 2: “Full emulation”[8]

It seems suitable to assume that actual hardware emulators are (though not necessarily) harder and more expensive to build than to program a software substitute for the piece of hardware to be emulated. Perhaps, as hardware synthesis methods improve, hardware emulation might become competitive with the software-emulation-of-hardware approach discussed above.

3.2 Operating system emulation

One can easily see that emulating between operating systems (cross platform emulation) can serve a wide range of purposes. These reach from emulating obsolete gaming platforms to historic computing to simply gaining some key features from older operating systems by imitating them on a current OS.

In order to run a random obsolete program on a “new” computer, one first has to create the original environment. An operating system is the interface between computer hardware and a user. Accordingly, when attempting to emulate one, one has to consider certain significant interactions such as visual displays, sound- and network capabilities and perhaps even varying file systems. The interfaces towards operating systems need to be re-implemented in some cases where the OS communicates with the underlying hardware in a totally different way. A common example is that Microsoft’s current operating systems use the NTFS-file system whereas earlier in history it was the standard File-Allocation-Table system (FAT).[2] This has to be considered when emulation earlier Windows versions.

In addition, the target and host platforms have to be analyzed. By "target", the platform on which the original objects run is meant. In case it is intended to reproduce 100% of the old operating system's behavior, one has to re-implement the systems technical features as stated in its source code, considering some adjustments as mentioned in the paragraph above. Otherwise, previously limiting the range of objects to be used by the emulated platform determines the required characteristics of a minimal target platform. Dealing with the host platform, there are two distinguishable approaches on how to run the emulator. The first and most straight-forward way is to run it directly on the host platform, giving away the highly desirable feature of portability. This way, the created emulator can only be run in one specific operating system. Second of choice is to employ a virtual machine to act as a "mediator" between emulator and host platform.[8]

3.3 Software emulation

When applying the principle of emulation to software, one can easily see that this barely has anything to do with preservation. Since most presently used programming languages often build on one another, there is no need to emulate between languages unless they are of different types like Java, C or Visual Basic for instance. In fact, software emulation is used in software engineering projects. When it comes to looking for software errors, the use of so called "fault injection" to emulate the effects of real software faults has been recognized as potentially very useful. Within this process the target program code is enriched with small changes thus emulating the original piece of software. This way the behavior of a software system can be analyzed in the presence of errors, making it more reliable once unwanted exceptions are fixed and the injected faults are removed again.[2]

For instance, when injecting faults in a device driver, one can evaluate the operating system's behavior in presence of a bad driver. On the other hand, when flawing the operating system, one can evaluate the fault-tolerance of the mentioned device driver.[2]

[2]http://technet.microsoft.com/en-us/library/cc758691(WS.10).aspx

4 Current projects on emulation

4.1 The CAMiLEON Project

The "CAMiLEON Project" is developing and evaluating a range of technical strategies for the long term preservation of digital materials. It is a joint project between the Universities of Michigan (USA) and Leeds (UK). Therefore CAMiLEON stands for "Creative Archiving at Michigan & Leeds: Emulating the Old on the New". Its primary objectives are:[10]

- To explore the options for long-term retention of the original functionality and "look and feel" of digital objects.
- To investigate technology emulation as a long-term strategy for long-term preservation and access to digital objects.
- To consider where and how emulation fits into a suite of digital preservation strategies.

There are three key strategies to the CAMiLEON Project.

Two of these are called "Software Longevity" and "Migration on Request", which both follow the migration and UVC strategies and are therefore not to be explained here.

The second strategy is emulation, which is to hold great promises for the future according to [10]. In fact the approach used in this case is also called "migrated emulation". The following graphic shows the intention to rebuild emulators from time to time, having the positive side effect of constantly having no more than one layer between host platform and digital document. Hence, whenever the old operating system becomes obsolete and new one is current, the emulator is translated by a compiler to run in the new environment.

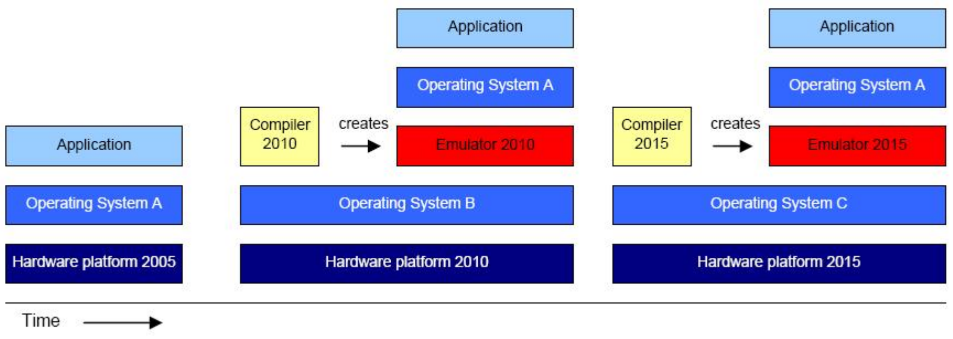

Figure 3: Migrated emulation via compilation[8]

4.2 The DIAS solution at the Koninklijke Bibliotheek

The “Koninklijke Bibliotheek”/National Library of the Netherlands (KB) has developed a specific workflow for archiving electronic publications. The technical heart of the e-Depot system is IBM’s “DIAS” (Digital Information and Archiving System). It includes import and export interfaces for different clients. These components (Delivery & Capture and Packaging & Delivery) are illustrated below.[9]

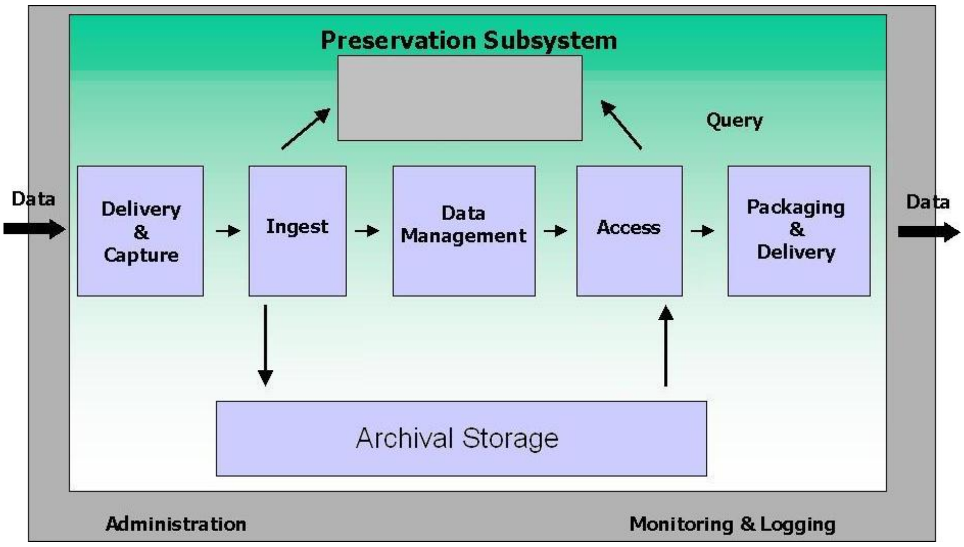

Figure 4: DIAS framework[3]

While the Delivery & Capture takes care of the pre-processing of digital objects, the Packaging & Delivery transforms the requested documents into an accessible format to the client.[3]
The actual detailed processes within DIAS shown in the picture above are of lesser importance since preservation strategies change quickly over time. In fact, a preservation project called “PLANETS” is currently being worked on using the DIAS framework described above.[3]

4.3 Dioscuri - The modular emulator

The “Dioscuri” hardware emulator is a result of the PLANETS preservation project within the framework of the KB e-Depot preservation research program. Its two key features are durability and flexibility. Because it is implemented in Java, it can be ported to any computer platform which supports the Java Virtual Machine (JVM), without any extra effort. It

[3]hhttp://www.kb.nl/dnp/e-depot/dm/dias-en.html

seems unlikely that emulation will fail since Java is a relatively wide-spread programming language and is supported by nearly all current computing platforms. Durability is achieved by the fact that Dioscuri is completely component-based. Combining multiple modules, which are software surrogates of hardware components, allows users to configure any Intel computer system (Dioscuri is based on the x86). Obviously these components have to be compatible. Otherwise one has to replenish Dioscuri's software library with updated or new modules. The figure below shows the simplified design of the emulator.[4]

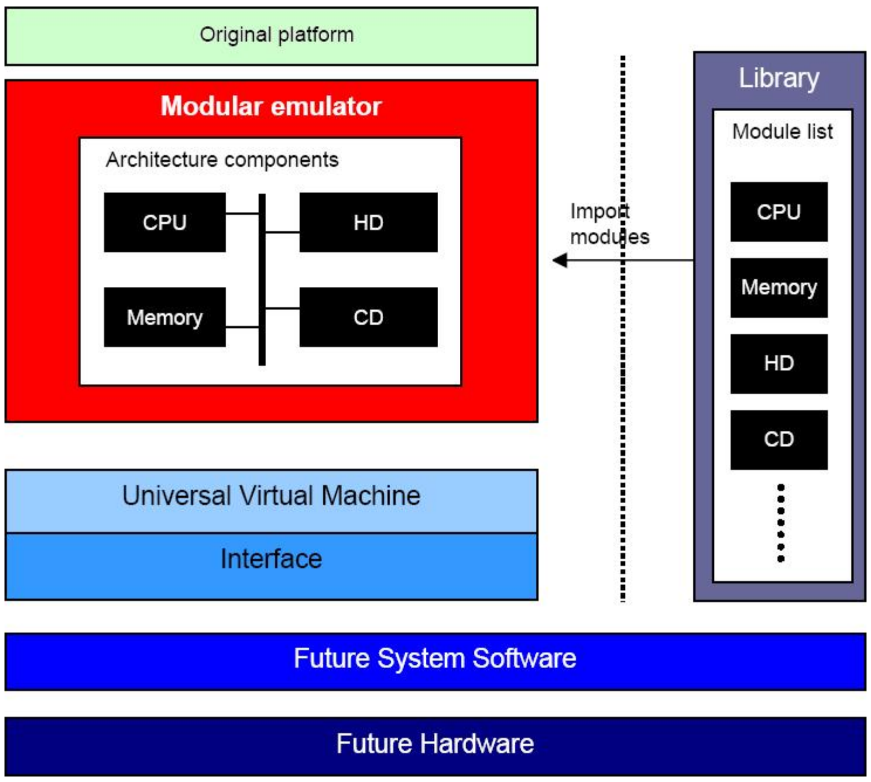

Figure 5: Dioscuri design[8]

Obviously, the emulator is the core element of the design. Its task is to virtually recreate the hardware of the upper (emulated) platform in such a way that the original pieces of software can run on it as they have on the original real hardware. The modular emulator feeds from a given library as shown on the right-hand-side. [4]

[4]http://dioscuri.sourceforge.net/dioscuri.html

5 Conclusion & Outlook

Having summarized the strategy of emulation for long-term preservation, some concluding words shall follow. In principle, emulation can be considered as the primary preservation strategy due to its unique way of preserving authenticity.

Starting with the CAMiLEON project, one can see that it does not actually put that much emphasis in emulation. Nevertheless, its focus is still on keeping the 'look and feel'. Dioscuri on the other hand makes use of emulation in a way that it can behave like a lot of older Intel computers. All things considered, the Dioscuri emulator seems most suitable for long-term preservation since it was particularly designed for this. The only maintenance effort one has to carry out is to port the actual emulator software once in a while and to keep the encapsulated documentation human-readable. Thus, as long as obsolete operating systems and software are stored safely, there should be no problem to make them run in this modular emulator.

There is another innovative emulation project which has just started in 2009. "KEEP" (Keeping Emulation Environments Portable) is a medium scale project. The development is supposed to provide accurate rendering of static and dynamic digital objects, reaching from simple text files to complex databases and videogames. The central goal of the KEEP project is to provide new tools for accessing any digital object both at present and in the long term. The illustration below shows a schematic decomposition of all steps in the process of the emulation access platform. The design is based on a digital data carrier as input A and possible outputs B and C offering two different approaches for representing the original digital document that was captured on the original carrier.

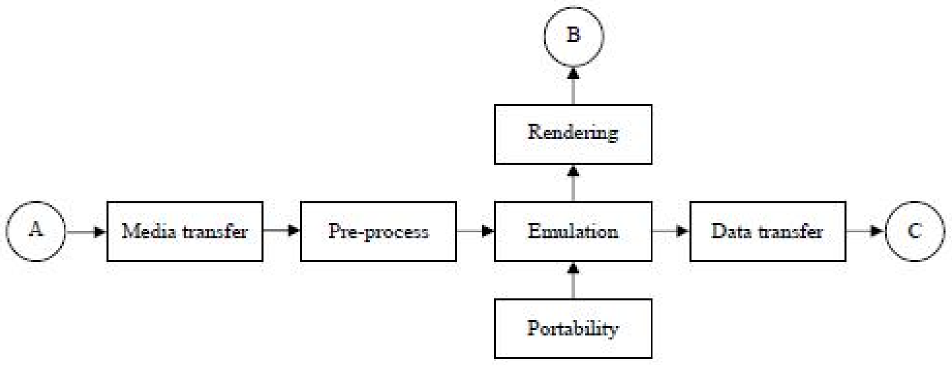

Figure 6: Decomposition scheme of KEEP[5]

With media transfer, the transport from the old data carrier to the target medium is meant. It is of utmost importance that the original bit stream is left unchanged. The next step is where KEEP's contribution is to be employed, since a robust and standardized data format is needed to store the scanned documents safely. The emulation part is ideally to be done by emulators that were created for the purpose of long-term preservation such as Dioscuri (see Chapter 4.3). As far as rendering is concerned, KEEP aims to ease the setup and operation of emulated environments by using a browsing system which searches the internet for metadata of a digital object.[5]

[5]http://www.keepproject.eu/ezpub2/index.php?/eng/About-KEEP/Technical-solution/Progress-beyond-the-state-of-the-art

References

[1] U.M. Borghoff, P. Rödig, J. Scheffczyk, L. Schmitz, 2003, *Long-Term Preservation of Digital Documents.* page 33, ISBN-10 3-540-33639-7 Springer Berlin Heidelberg New York.

[2] J. Duraes, H. Madeira, 2002, *Emulation of software faults by educated mutations at machine-code level.* Software Reliability Engineering, 2002. ISSRE 2002. Proceedings. 13th International Symposium on

[3] Adam Farquhar, 2006, *Planets - Preservation and Long-term Access via NETworked Services.* available at: http://www.planets-project.eu/docs/comms/Planets_Project_Brochure.pdf

[4] Jens Frauenschläger, 2002, *Hardware-Debugging durch die Kombination von Emulation und Simulation.* page 14, Diplom thesis at "Universität Leipzig", available at: http://lips.informatik.uni-leipzig.de/files/2002-58.pdf

[5] Soha Hassoun, Senior Member, IEEE, Murali Kudlugi, Duaine Pryor, and Charles Selvidge, February 2005, *A Transaction-Based Unified Architecture for Simulation and Emulation.* IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 13, NO. 2

[6] Ir. Raymond Lorie, 2002, *The UVC: a Method for Preserving Digital Documents - Proof of Concept.* IBM Netherlands, Amsterdam / KB Long-Term Preservation Study

[7] Jeff Rothenberg, January 1998, *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation.* Chapter 8, Council on Library and Information Resources, Commission on Preservation and Access

[8] Jeffrey van der Hoeven, Hilde van Wijngaarden (KB), 2005, *Emulation - a viable preservation strategy.* Nationaal Archief of the Netherlands, available at: http://www.kb.nl/hrd/dd/dd_projecten/-Emulation_research_KB_NA_2005.pdf

[9] Hilde van Wijngaarden, Frank Houtman, Marcel Ras, 2008, *The KB e-Depot in development Integrating research results in the library organisation.* page 3, iPRES 2008 (British Library Conference Centre) available at: http://www.bl.uk/ipres2008/presentations_day2/41_Ras.pdf

[10] Paul Wheatley, 2001, *Migration - a CAMiLEON discussion paper.* available at: http://www.ariadne.ac.uk/issue29/camileon/

[11] IBM Corporation, *IBM System/360 Principles of Operation.* 2003, Form A22-68-21-0, available at: http://bitsavers.org/pdf/ibm/360/princOps/A22-6821-0_360PrincOps.pdf