Meeting Digital Preservation Requirements for Software through an Emulation Strategy

Christophe Ponsard
CETIC Research Centre, Charleroi, Belgium

Keywords: Digital Preservation, Preservation Strategies, Emulation, Migration, Virtual Machine, Retrocomputing.

Abstract: Digital preservation aims at ensuring digital artefacts remain accessible and usable. This includes preserving application software for the resulting experience but also software required to provide access to valuable digital artefacts. This paper surveys different preservation strategies of such software with a focus on the use of emulation which is gaining momentum over the more traditional migration approach. We highlight some requirements to consider when selecting emulators. We illustrate the process on the preservation of software on a micro-computers of the 1980's. We also discuss how to design software architectures for the long term preservation of the emulators themselves.

1 INTRODUCTION

Over the past 50 years, our world has experienced a digital revolution with the exponential development of digital processing capabilities as reflected by Moore's "Law". Digital information has followed a similar evolution with the emergence of Big Data.

Digital preservation can be defined as *"the processes aimed at ensuring the continued accessibility of digital materials. To do this involves finding ways to re-present what was originally presented to users by a combination of software and hardware tools acting on data."* (UNESCO, 2003). It concerns materials that are born digital or (partial) digital representation of physical objects through the use of scanning or recording devices. Many actors are concerned from curators in digital(ized) libraries or cultural institutions, public institutions, private companies and even at a smaller scale family heritage.

Digital preservation is a complex issue because information is immaterial and requires to consider the preservation of the whole chain to allow this information to be re-experienced and re-interpreted by a human brain. This includes (Rauber, 2013):

- *physical objects* such as the media (tapes, disks, solid state storage,...) which need to be preserved from physical damage.
- *logical objects* (software, files) which are processed by a machine following specific execution or encoding format which must be preserved.
- *conceptual objects* having a meaning to human, in some language and culture, e.g. a document mixing pictures and comments representing a photo album.

In the scope of this paper, we focus on a specific kind of digital artefacts: application software of historical value, typically quite early software (including games) produced in the past century. The motivation to preserve such software might fall in different scenarios such as:

- *data recovery*, for accessing historical data that was not migrated to newer platforms.
- *nostalgia*, often for experiencing again old software (retro-computing) or games (retro-gaming).
- *real usage*, for programs that are not available any more, e.g. some writers prefer pure text processors over WYSIWIG ones.
- *historical*, for studying the design of old computer systems.

This paper presents our work in progress to explore the use of emulators for this digital preservation context. Emulators have a long history depicted in the timeline of Figure 1 and have been used for compatibility, research, cost reduction and more recently for digital preservation purpose (Rosenthal, 2015).

The structure of this paper reflects our research methodology and is structured as follows. First, in Section 2, we motivate the choice of an emulator-based strategy by anchoring it in our requirements and making a comparative analysis together with

Ponsard, C.
Meeting Digital Preservation Requirements for Software through an Emulation Strategy.
DOI: 10.5220/0010602903180323
In *Proceedings of the 16th International Conference on Software Technologies (ICSOFT 2021)*, pages 318-323
ISBN: 978-989-758-523-4

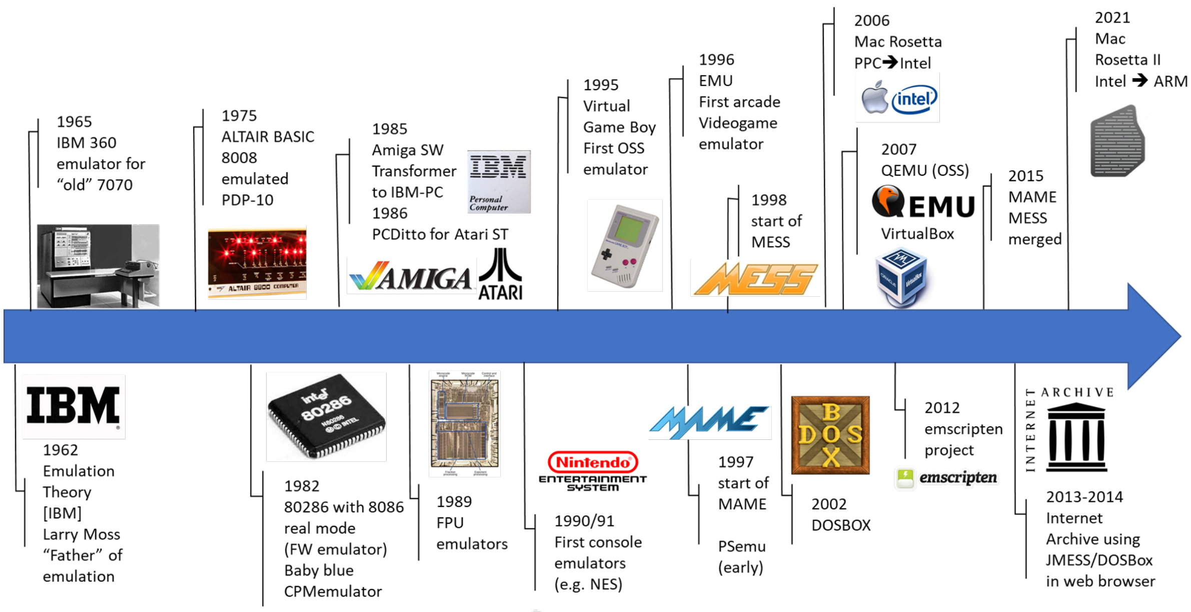


Figure 1: Emulator timeline (main milestones).

other strategies such as standardisation, migration, encapsulation or universal virtual machine (Rauber, 2013)(Shimray and Ramaiah, 2018). This work relies on a literature survey of different authors with different opinion about emulation (Granger, 2000)(Lee et al., 2002)(Cochrane et al., 2018). Then, Section 3 shows how to develop an emulator-based solution by identifying important criteria and detailing how to select one or multiple emulator(s) matching the desired criteria. The importance of building a consistent toolchain is also discussed. This part is illustrated on two complementary case studies of micro-computers from the 1980's. The last part of our research is discussed in Section 4 and covers longer term architectural considerations about the preservation of the emulators themselves considering the evolution of their host. Finally, Section 5 draws some conclusions and presents our future work.

2 MOTIVATION FOR AN EMULATION APPROACH

2.1 Requirements for Software Preservation

Our context involves a number of specific requirements and constraints that must be met by the considered strategy:

- it is about software, so it requires some form of execution which needs to be close enough to the original experience. Differences such as degradation (loss of features) or enhancements (e.g. improved screen rendering) must be traced.
- original hardware might be rare or subject to usage restriction (typically in museum which bother first to preserve the physical object).
- early hardware or software might be quite specific and non-standard so require quite specific knowledge to operate them.

2.2 Overview of Digital Preservation Strategies

Two main digital strategies are usually considered migration and emulation (Hoeven et al., 2007)

- *Migration* focuses on the digital object itself. It aims at transforming the object in order to be able to preserve its conceptual representation on the new platform. Typically, a document will be migrated to a more recent format. The threat is the progressively degradation if format conversion is not perfect. Very old formats might also be lost and require reverse engineering.
- *Emulation* does not change the digital object, but aims at reproducing the hardware and software environment required for rendering the original object. An emulator is an hardware or software that enables one computer system (called the host) to behave like another computer system (called the guest). The threat here is the inability to reproduce this environment or not reaching enough

accuracy. Some legal basis may also be required. Other strategies may also be considered (Rauber, 2013)(Shimray and Ramaiah, 2018), notably:

- *Standardisation* for stability and interoperability.
- *Extraction*, to recover partial information.
- *Encapsulation*, to fully document an environment.
- *Universal Virtual Machine*, to rebuild a full environment from scratch.

2.3 Comparison of Digital Preservation Strategies

Considering the requirements described in Section 2.1 ans strategies identified in Section 2.2, we performed a comparative analysis. Table 1 summarises all the strategies, illustrates typical usage scenarios which can be compared to our scenarios and identifies some problems that need to be addressed or could rule out a strategy if not possible.

In our case, many approaches are not suitable. First of all, total preservation is denying the evolution and is very expensive. Then, standardisation only developed with time and many old machines have specific design and lack compatibility which could be used to run the program on later machines. Moreover standards are also not supported forever and more generally backward compatibility is often dropped after a few versions so it needs to be combined with migration. About encapsulation, it helps provisioning for the future through capturing information, e.g. specific dependencies important for running the software, but it is only part of a solution. Finally, migration is more suitable for static (data) artefact than for software which is highly dynamic.

In the end, emulation emerges as one of the major solution but needs to cope with some problems such as emulator accuracy and development/maintenance. Those points are respectively addressed in Section 4 and 5. There are also possible legal issues related to emulation (e.g. copyrighted material such as ROMs or the notion of abandonware) which are not discussed in the scope of this paper.

A final approach worth investigating is the Universal Virtual Machine which recreates the environment for accessing a specific document format. The key idea is to keep such a platform easy to implement on any host system and to run a generic document viewer on top of it. This approach is considered for the emulator artefact itself in Section 5.

3 BUILDING A DIGITAL PRESERVATION TOOLCHAIN

3.1 Emulator Selection Criteria

Selecting a suitable emulator depends on different needs which may vary depending on the context. Table 2 details a number of relevant criteria to consider such as the ease of installation and configuration, the

Table 1: Comparison of Digital Preservation Strategies.

Method	Idea	Example	Problem
Total preservation	Keep everything in state	Recap computer, change drive belt, refresh floppy disks...	Costs/expertise for old HW, old media Restricted usage
Standardisation	Standard are there for a long time	Running on any IBM PC compatible	Few standard in retro time How long is “a long time”
Encapsulation	Container with useful meta-data, (links to) software	Format information to decode image Instructions how to run a program	Documentation, access Not solving anything
Extraction	Mining useful stuff	Text without images, decompilation	Partial, fallback in degraded mode
Migration (media)	Transfer to more stable/accessible media	From tape/floppy to HD/SSD/Cloud (wav/binary files)	Lost media specificities (or use disk image) Lifespan of new media ?
Migration (backward compatibility) Interoperability	Can read/run previous version on current applications	Open doc/check/save Run win32 application on win64	Targeting documents Possible loss, progressive degradation Limited in time
Emulation	Keep digital resource in original form but emulate hardware	MAME for games MESS for old micro-computer DosBOX for DOS programs	Need to write/maintain emulator How to access data? How accurate is the emulation? Possible legal issues
(Universal) Virtual Machine	Ensure independence w.r.t. host platform using simple VM	Historical UVM concept Concrete: Java or Javascript?	UVC targeting documents not programs

Table 2: Emulator Selection Criteria.

Criteria	The bad	The good	Hints
Ease of installation and configuration	Need to compile, find ROM, configure keyboard,... Platform bundles/pre-configured	Running in browser [e.g. Internet Archive]	Multi-system more difficult Easy All-in-one distribution on Raspberry Pi
Ease of use	Raw emulator, poor media management	Power utilities integrated (media mgt, snapshots, debugger,...) Nice front-end	Machine/constructor specific can bundle utilities Providing libretro API
Accuracy	Abstraction of many components	Circuit/Cycle exact (but higher CPU)	Check forum, game compatibility
Long term support	Recent project closed or small community exotic technology	Long history large community VM approach	Check repository activities and OpenHub statistics

ease of use, the emulator accuracy (which may impact the resource required), its long term support, and required resources. For each criteria, some benefits and drawbacks are identified. Some hints are also proposed in order to guide in the assessment of the criteria. For example, specific software (such as highly optimised games) may require high accuracy such as cycle exact emulation. In order to help assessing the maturity and support for an open source emulators, open source monitoring website such as OpenHub are also helpful (Black Duck, 2006).

Note that the emulator choice does not need to be restricted to a single solution. Actually complementary emulators may be selected as long as they can integrate in a common toolchain, for example through the sharing of disk images. This can allow the use of a specialised emulator for specific scenarios, for example a more accurate emulator can complement a more generic one for running some tricky software written in assembly language at the price of more computing power and maybe also more configuration time. As more emulators means more maintenance work, it is recommended sticking to a minimal number.

3.2 Example of Emulator Selection

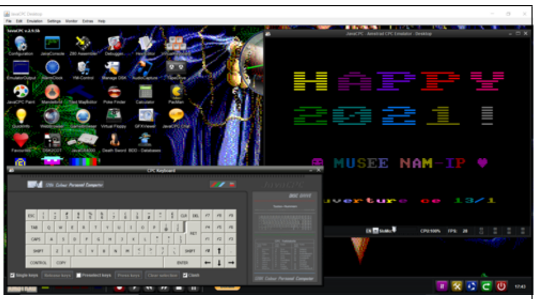
In order to illustrate the selection process, we take the case of the Amstrad CPC (covering CPC 464, 664 et 6128 machines and "plus" variants, depending on memory and tape vs disk). Those 8-bits machines are based on a Z80 CPU and were quite widespread. There are more than 30 emulators (including variants but not considering browser ports) available from multi-system platforms, such as MAME, to totally specific emulators such as JavaCPC. We will use those two extremes as candidates and compare them using the list of criteria detailed in Table 2.

Table 3 summarises our comparison work and also illustrates the respective emulators running the same BASIC program using the integrated ROM interpreter. The program was loaded using the same disk image showing the interoperability between the emulators. The installation was largely more easy with JavaCPC as long as a JVM is installed it will directly boot and provide a full desktop interface with advanced utilities such as bidirectional copy/paste, integrated debugger with breakpoints or a virtual line printer. On the other hand, MAME required to find a ROM and to go through a more painful keyboard configuration. Looking at the support, MAME is actively maintained while JavaCPC has not been active for many years according to OpenHub. Emulators produced by computer fans may face that risk. In this case, it is mitigated by the fact the project is Open Source and compiled to JVM bytecode meaning it can run on many platforms supporting Java. In contrast, MAME offer more guarantees of long term support. In the end, using a combination of emulators is interesting as JavaCPC provides a complete toolchain at no cost while MAME guarantees a longer support. Many other features are common such as disk and tape emulation, snapshots or accelerated modes.

3.3 Elaborating a Complete Digital Toolchain

Depending on the emulator, more or less utilities might be available in order to provide all the useful integration capabilities with physical or virtualised devices such as tape or disk storage. It is interesting to build a graph of dependencies between various utilities to check if there is a good interoperability between physical and emulated devices. In the case of our CPC, but also for many other micro-computers, disk images in a standard format can be exchanged across emulators. They can also be installed on disk emulator directly connected to the physical interface to experience the software on the real hardware.

Table 3: Comparison of the Amstrad CPC emulator: MAME vs JavaCPC.

	MAME CPC emulator	JavaCPC emulator
		
Type	Generic emulator	More specific (based on JEMU but only limited targets)
Ease of use	Need to install ROMS, configure keyboards,...	Lot of features, everything preconfigured, copy/paste, printer
Accuracy	OK in Basic (not assessed further)	OK in Basic (not assessed further)
Long term support	Open Source Metrics (OpenHub/MAME) ... has had 77,136 commits made by 498 contributors representing 10,529,234 lines of code ... is mostly written in C++ with an average number of source code comments ... has a well established, mature codebase maintained by a very large development team with stable Y-O-Y commits ... took an estimated 3,239 years of effort (COCOMO) starting with its first commit in December, 2007 ending with its most recent commit about 1 month ago	Open Source Metrics (OpenHub/JavaCPC) ... has had 30 commits made by 3 contributors representing 38,957 lines of code ... is mostly written in Java with a low number of source code comments ... has a codebase with a very short history maintained by nobody with stable Y-O-Y commits ... took an estimated 10 years of effort (COCOMO) starting with its first commit in December, 2008 ending with its most recent commit over 11 years ago

4 LONG TERM PRESERVATION OF EMULATORS

An emulator runs on an host platform which is subject to evolution. In the event of its replacement by another platform, the way to keep the system operational must be investigated. In the literature, we could identify strategies for long term evolution which can also be mixed: emulator recompilation, stacking or evolution through a virtual machine (Van Der Hoeven and Van Wijngaarden, 2005)(Rosenthal, 2015).

4.1 Emulator Recompilation

The first strategy is illustrated in Figure 2. In order to cope with the API of the new operating system, the emulator needs to be recompiled against this system using a compiler which is also running on that new system. For mainstream compiler such as C/C++ used by multi-emulator systems, this can be a reasonable assumption. The compilation process may need some tuning in case of specific evolution and thus may require support from the emulator community. It is also recommended to have good test suites in order to make sure the new port is fully qualified. The result is a new emulator running directly on top of the new OS thus with optimal performance.

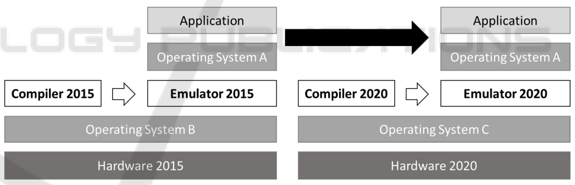


Figure 2: Emulator timeline (main milestones).

4.2 Emulator Stacking

The second strategy avoids to recompile the emulator. Consequently, it requires the old platform to be itself emulated on the new platform, resulting in a stack of emulators as depicted in Figure 3. This strategy does not put responsibility on the emulator community. It is more likely to the developer of the operating system to provide backward compatibility in case of platform evolution, e.g. in case of substantial hardware change. An example is the Apple Rosetta emulators to cope with migration to Intel in 2006 and to ARM, more recently. However, such support may be limited in time and other strategies may eventually need to be considered. Note this stacking comes at the price of reduced performance. The degradation will also worsen as the stack is growing, although this can be compensated by the continuous increase in processing power we have experienced over the past 50 years.

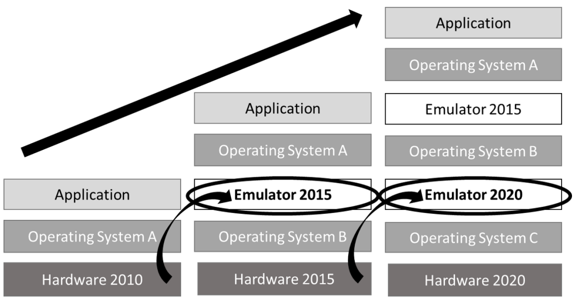


Figure 3: Emulator timeline (main milestones).

4.3 Virtual Machine

A third strategy is to make the choice of a virtual machine to isolate the emulator application from the real host. This will greatly ease the evolution because only the virtual machine needs to be ported to the new target host as depicted in Figure 4. The virtual machine may be a standard technology maintained by a third party which then relieves the community from any recompilation work. There are some possible issues in the evolution of the virtual machine itself over time, although probably at a slower pace that the evolution of the guest Operating System. Nevertheless, backward compatibility must be preserved, otherwise a recompilation strategy can become necessary.

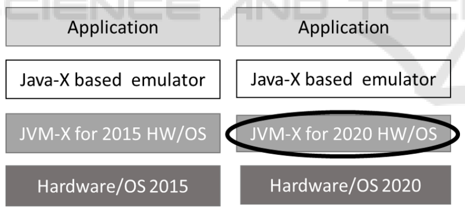


Figure 4: Emulator timeline (main milestones).

4.4 Mixed Approaches

Different approaches may be combined in various ways. Stacking may be considered temporary as inefficient solution waiting for a recompilation or the availability of the port of a virtual machine. Recompilation may also be considered later in order to fully realign with a target platform and improve efficiency.

5 CONCLUSION & NEXT STEPS

In this paper, we have motivated the use of emulation to ensure digital preservation of software artefacts based on specific requirements and usage scenarios. We have also identified key criteria to guide the selection of an emulator and illustrated the process on a concrete case. The choice may not be restricted to a single emulator and needs to integrate in a full digital preservation chain. Finally, we have also discussed different strategies for the longer term evolution of the emulator themselves which can also rely on virtual machines.

As future work, we plan to elaborate our emulator selection criteria, to provide more guidance and to expand our study to cover a wider set of scenarios going beyond pure digital preservation, e.g. also considering interoperability requirements. On the technological side, our intent is to investigate more recent emulation technologies and also DevOps techniques for automated regression testing of emulator builds.

ACKNOWLEDGEMENTS

Thanks to the NAM-IP Computer Museum (Belgium) for access to its collections.

REFERENCES

Black Duck (2006). Openhub. https://www.openhub.net.

Cochrane, E., Tilbury, J., and Stobbe, O. (2018). Adding emulation functionality to existing digital preservation infrastructure. *Journal of Digital Media Management*, 6(3).

Granger, S. (2000). Emulation as a Digital Preservation Strategy. *D-Lib Magazine*, 6(10).

Hoeven, J., Lohman, B., and Verdegem, R. (2007). Emulation for digital preservation in practice: The results. *International Journal of Digital Curation*, 2.

Lee, K.-H., Slattery, O., Lu, R., Tang, X., and McCrary, V. (2002). The state of the art and practice in digital preservation. *Journal of Research of the National Institute of Standards and Technology*, 107.

Rauber, A. (2013). An Introduction to Digital Preservation. IFS, Vienna http://www.ifs.tuwien.ac.at/~andi.

Rosenthal, D. S. (2015). Emulation & Virtualization as Preservation Strategies. report commissioned by The Andrew W. Mellon Foundation https://mellon.org/Rosenthal-Emulation-2015.

Shimray, S. and Ramaiah, C. (2018). Digital preservation strategies: An overview. In *Proc. of 11th Conference on Recent Advances in Information Technology*.

UNESCO (2003). Concept of Digital Preservation. https://unesdoc.unesco.org/ark:/48223/pf0000130071.

Van Der Hoeven, J. and Van Wijngaarden, H. (2005). Modular emulation as a long-term preservation strategy for digital objects. In *5th Int. Web Archiving Workshop*.