Evaluating Emulation and Migration: Birds of a Feather?

Mark Guttenbrunner and Andreas Rauber

Secure Business Austria, Vienna, Austria
Vienna University of Technology, Vienna, Austria
{mguttenbrunner,arauber}@sba-research.org

Abstract. Evaluating the results of a digital preservation action, be it migration or emulation, is a complex task. The usual approach for migration is to evaluate object properties after the migration. For emulation strategies the result of the rendering of the object is evaluated. In this paper we argue that the change of object properties when migrating is not sufficient evidence if a digital preservation action is successful or not. Even for migration the rendering process of the digital object is crucial, and as such evaluating object properties is not enough. The difference in evaluation between emulation and migration as a strategy for digital preservation becomes blurred as migration results have to be compared based on the rendering of the target format and the environment used to render the migrated digital object. Evaluation of object properties when migrating will only validate a necessary condition for preserving significant properties, i.e. whether the information underlying a specific property is still present in an object after migration. It cannot guarantee that the rendering based upon the migrated object will exhibit a specific significant property. In this paper we show the view-path of digital objects and explain how emulation and migration actions affect it. We then compare the changes that occur in the view-path and show that these are at least as severe when migrating a digital object as when emulating its rendering environment.

1 Introduction

Emulation and migration as the main strategies in digital preservation are usually treated as entirely different strategies. While for evaluating the success of a migration action the object properties are compared to check if significant properties of the object change, the rendering environments of the digital object (i.e., the environment the object was originally rendered in and the environment it will be rendered in after the migration) are quite frequently not taken into account. With emulation, on the other hand, the digital object does not change, so only the rendering of the digital object in the original environment and the rendering in the emulated environment are compared to see if the rendering is identical.

What is usually not considered in the evaluation of a migration action is that every extraction of significant properties of an object is already a form of rendering i.e. interpretation of this object. Even though not necessarily directly visible to the user, the object is rendered by the routines used to extract the properties. The problem with this approach is that the program ”rendering” the object is neither necessarily the program originally used to render it nor the one that will be used to render it and thus the results are not necessarily authentic to the original rendering once the object is rendered in a different environment. (Note that ”rendering” in this context is not restricted to the visual display of an object. It refers to all kind of interpretations of an object and the resulting effect on an environment, be it visual, acoustic or effects on a system state, files stored on media, or communication/voltage levels on I/O ports, etc.)

In this paper we will show how migration and emulation strategies affect the view-path of a digital object on different levels. By applying a digital preservation action the view-path is changed, both for emulation and migration actions. The rendering using the new view-path has to be compared to the object rendered in the original view-path to evaluate changes of significant properties of the object when rendered in the new view-path. While this is common knowledge when evaluating emulators this paper will emphasize the similarities in rendering of a migrated object and the changes in the view-path. Both the evaluation of a migration action and an emulation action should thus always include the combination between object and rendering environment.

This paper is structured as follows. First, we provide an overview of work relevant for this paper. In Section 3 we examine the generic view-path of rendering any digital object and describe different layers that appear in the view-path and on what levels emulation can take place. Next, we show how the view-path changes with emulation on the different levels in Section 4. In Section 5 we then explain how the view-path changes when migrating a digital object. Next, we compare the resulting view-paths after migration and emulation and argue why there should be no difference in evaluation between migration and emulation. Finally, in Section 7 we present our conclusions and give an outlook to future work.

2 Related Work

Rothenberg argues, that „Digital Informational Entities are Executable Programs” [7], i.e. that every digital object is a program that has to be interpreted by a process that knows how to perform the commands in the formal language (the format) the program is written in. This can be as simple as interpreting a string of ASCII character codes to make it human readable. Most of these interpreters are software, but on the lowest level the machine-code is interpreted by hardware (i.e., the CPU of a system).

This stack of interpreters used to decode a digital object is called the view-path. Van Diessen describes it as „a full set of functionality for rendering the information contained in a digital object” [10]. The view-path contains the hardware and all the secondary digital objects needed to render an object along with their configuration. It is possible to use different view-paths to display the same object (e.g., different viewer applications that can render the same digital object).

For digital preservation purposes it is necessary to compare the effects of different preservation actions on the significant properties of digital objects. When migrating a digital object to a different format, characteristics of the object are usually extracted from both file representations and compared. Becker et. al. show a preservation planning workflow that allows for repeatable evaluation of preservation alternatives [1]. This workflow is implemented in the preservation planning tool *Plato* [2]. Part of preservation planning is to automatically characterize migrated objects using tools like Droid to identify files [4]. Migration results can be validated automatically supported by the eXtensible Characterisation Languages (XCL) [3]. The original and migrated objects are hierarchically decomposed and represented in XML. These representations can be compared to measure the effects of the migration on the digital object on properties extracted from the object.

Thaller suggests to separate the information contained within a file from the rendering of the information [9]. Information stored in the file can, for example, be the coordinates of text which leads to the rendering displaying the text on a specific point on the screen. This is described as the *look & feel* aspect of an object. Case studies of interactive objects comparing the rendering outcomes of different rendering environments using the aforementioned characterization language XCL on the level of screenshots of renderings have been carried out [6].

We previously introduced a conceptual framework on how to evaluate rendering environments [5]. While the paper mainly concentrates on emulation, the presented concepts can be applied to all kind of rendering environments. This paper will further strengthen the argumentation that the evaluation of rendering environments is necessary both for migration and for emulation.

3 View-Path and Levels of Emulation

The view-path is the stack of objects needed to render a digital object. Even in the simple case of rendering a static document at least an application, the operating system and the hardware needed to run the operating system will be present in the view-path. An example would be a PDF-A 1.1 document rendered using Adobe Acrobat Reader 10.1.3 on a Microsoft Windows 7 operating system updated to a certain date/service pack and a specific set of fonts installed and specific language settings on an x86-compatible workstation PC hardware with a specific graphics card, sound chip etc. A generalized version of this simple view-path can be seen in Figure 1a.

Depending on the digital object some of the layers in the view-path can be missing or additional layers can be present. A computer game would usually run directly on the operating system (or on more ancient systems directly on the hardware), while an application written in Java will additionally have a specific Java Virtual Machine in its view-path.

While emulation in digital preservation is frequently seen as recreating the hardware of the system the digital object was initially rendered on and using the original software (both the operating system as well as the application) to

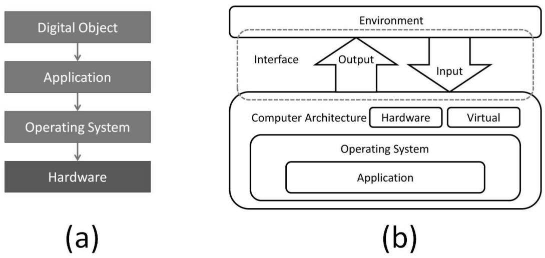


Fig. 1. Generic view-path for rendering a digital object (a), layers of emulated environments for digital objects (b)

display it, emulation can be done on different levels. Figure 1b shows the different layers in a system on which emulation can take place.

The term Emulation in general refers to the capability of a device or software to replicate the behavior of a different device or software. It is possible to use hardware to emulate hardware or to use software to emulate software or the hardware. Even a different viewer used for rendering a document emulates the functionality of the original viewer.

In this paper we concentrate on the layers inside a computer system, and differentiate between the following different layers on which functionality originally provided can be emulated by different rendering engines.

Application. As described in Figure 1a, an application is usually used to render a digital object (if the digital object to be preserved is not itself an application (e.g., computer games, digital art, self-running documents). By replacing the application used to render a digital object the functionality of the original application is emulated.

Operating System. On a modern computer system usually an operating system between the application and the hardware (virtual or real) provides functionality used by the application (e.g., access to display devices, input devices). The operating system, in turn, heavily depends on the computer architecture it is running on and the application heavily depends on the operating system. By providing a middle-ware that acts like the original operating system any operating system calls in the application can be caught by the middle-ware and translated for the new environment, thus emulating the functionality of the original operating system. An example for this would be Wine[1] that allows the execution of programs running on Microsoft Windows operating systems to be executed in a Linux environment.

Computer Architecture. As the functionality of hardware is usually better documented than software the most common level of emulation is recreating the computer architecture. Computer architecture can either be real hardware (an actual physically existing system) or a system only existing virtually, a virtual machine like e.g., the Java VM. While virtual machines are software that is usually running on top of the operating system and ported

[1] Wine - `http://www.winehq.org/`

into a different environment, physical hardware can be emulated using one of the following approaches:

Full Hardware Emulation. The most common use of emulation is the recreation of hardware components in software on a new host-system. The „Emulator" in this case is, as defined in [8], a program that runs on one computer virtually recreating a different computer's hardware. It provides a layer between the host system and the originally used software, replacing the functionality of the original physical hardware used to execute this software. The whole system is rebuilt in software and the original digital artifact is executed using the original software in this simulated hardware environment. Examples are emulators for proprietary hardware such as video game console systems on a PC or emulators for PC hardware on virtual machines (e.g., Dioscuri[2]).

Virtualization. An approach to create a virtual environment by using either some or all of the hardware of the host system directly is called virtualization. Code is directly executed on the physical CPU instead of being emulated. In reality the approach is usually a mix between hardware emulation and virtualization as all virtualization solutions emulate certain low-level instructions instead of running them directly on the real CPU. Using the virtualization approach, software which is potentially able to run on the host system's hardware is run in a virtual machine hosted by the current operating system environment. Examples for virtualization software are VirtualBox[3] and QEMU[4]. One use of virtualization is to have a host system running one operating system and create a virtualized environment running a different operating system. Another one is to run different virtualized computer systems on the same physical hardware in parallel. The long term use for digital preservation is obviously limited, as virtualization only works if the physical system uses a CPU that is compatible to the target system. As soon as the hardware becomes obsolete, computer architecture emulation has to be done using hardware emulation.

4 Changing the View-Path Using Emulation

Using emulation as a digital preservation strategy, we keep the object unchanged. This means we can replace three of the layers in the original view-path shown in Figure 1a. Based on the different levels of emulation, we have the following possible resulting view-pathes:

Application. The most simple level on which we can change the rendering is by replacing the application as seen in Figure 2b. An example for this would be to use FreePDF instead of Adobe Acrobat Reader to render a PDF

[2] Dioscuri - `http://dioscuri.sourceforge.net/`
[3] VirtualBox - `https://www.virtualbox.org/`
[4] QEMU - `http://wiki.qemu.org/Main_Page`

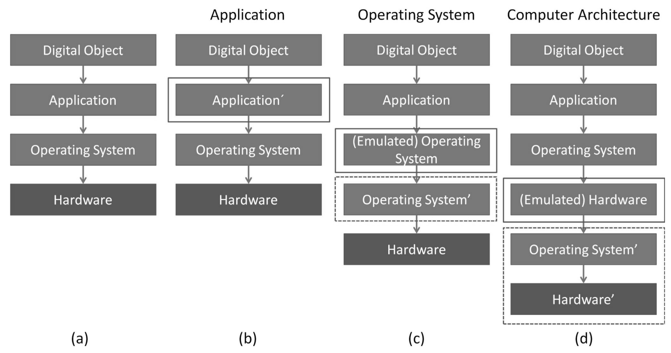


Fig. 2. Changing the view-path by emulating the application. Full boxes show the emulated layer that changes, dashed boxes additional layer(s) that need to be introduced. Shown are the original view-path (a) and view-paths emulating the application (b), the operating system (c), and the computer architecture (d).

document. While not a long term strategy (as the hardware and operating system stay the same), this could be the case if an application gets obsolete and needs to be replaced by a different one.

Operating System. The operating system is usually very closely tied to the underlying hardware. If we keep the original application in place in the view-path and use a translation layer emulating the originally used operating system, we get an extra layer in the view-path as shown in Figure 2c. As with the strategy of emulating on the application level, this is not a long term strategy, as the application would still need to be run on the same hardware. This strategy would be used if the operating system gets obsolete and needs to be replaced by a different one, still running on a compatible physical hardware.

Hardware. Finally we can replace the actual physical hardware by an emulator replacing its functionality. The original application and the operating system used to run the application stay unchanged. Introducing an emulator for the hardware means that usually two additional layers are introduced into the view-path as shown in Figure 2d: the emulator of the original hardware and the operating system on which the emulator runs. If the emulator is based on a virtual machine (e.g., Java VM), we get another additional layer. As in this approach the physical hardware is replaced by software that can be ported to new machines (the emulator), this is obviously also the most promising approach for a digital preservation strategy.

5 Changing the View-Path Using Migration

If we use migration as a digital preservation strategy, the digital object is changed. Based on the view-path shown in Figure 1a we get the following new possible view-path options.

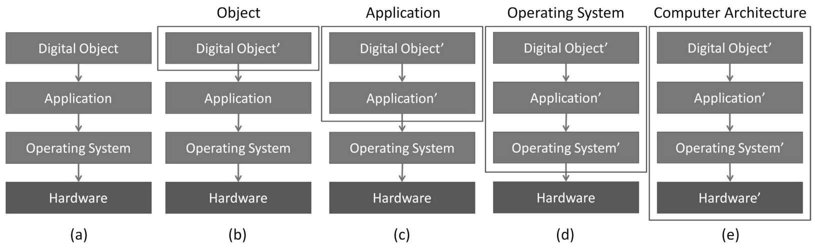


Fig. 3. Changing the view-path by migrating the digital object. Boxes around the layers highlight the layers that change. Shown are the original view-path (a) and the changed view-path when the object is migrated (b), when a different application is used to render it (c), when a different operating system has to be used for the application (d), and when the computer architecture changes (e).

Digital Object. If only the digital object is changed, but the same application is used to render the digital object as shown in Figure 3b, the remainder of the view-path remains unchanged. An example would be to convert an image in format BMP to PNG and use the same tool to view the image. The use as a strategy is limited as the motivation for a digital preservation is usually that the application used to render the digital object would get obsolete, and not just the format of the digital object.

Application. In addition to changing the digital object we may also need to change the application that is used to render the object. An example would be to migrate all documents from Microsoft Word DOC-format to PDF-A and using Adobe Acrobat Reader instead of Microsoft Word to render the objects. On this level the operating system and the hardware would still be unchanged as shown in Figure 3c. (Note that, strictly speaking, a new version of an application basically constitutes a new application that needs to be verified as such.)

Operating System. The operating system has to be replaced by a different one once it gets obsolete as shown in Figure 3d. Using a new operating system means usually to replace the applications (at least with new versions) as well. While it might still be possible to run it on the same hardware, this already changes 3 layers in the view-path. An example would be to migrate the object from Microsoft Word 95 format to Microsoft Word 2007 format, and at the same time change the application Microsoft Word 95 to Microsoft Word 2007 and the operating system from Microsoft Windows 95 to Microsoft Windows 7.

Hardware. As hardware gets obsolete as well, we finally will have to replace the actual physical hardware by a different one. The upgrade in hardware usually involves an upgrade in the operating system with all the before mentioned steps. As can be seen in Figure 3(e) this involves potentially to exchange all the layers in the view-path. Note, that we may find a situation where we replace (part of) the underlying hardware, but are still able to use, at least nominally, the same operating system. Yet, this usually needs to be treated like a complete replacement of also the operating system, and potentially any applications building on top of it as at least the operating system will usually not be identical: different drivers may be required to access different hardware components, API's behave differently, etc.

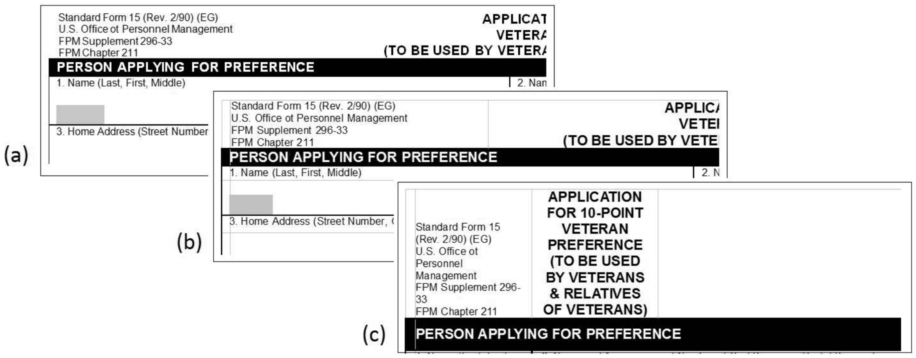
6 Comparison of the Strategies

Emulation strategies usually replace one layer in the view-path or add an additional layer as an interface between the emulated view-path and the new underlying layers. This adds complexity to the rendering of a digital object, with side-effects of the introduction of the new layers having to be considered. For evaluating the rendering we need to do a comparison between the rendering in the original view-path and the new view-path to judge where the rendering differs and if the digital preservation action is useful for the given setting. Any change in the view-path due to a layer getting obsolete has to lead to a re-evaluation of the rendering.

For migration the view-path of the digital object stays only the same if the same application on the same system is used. In every other case the view-path changes and every lower layer leads to all layers on top of it being changed as well. Still, as different code segments in the same application will be used to render an object (e.g. a PNG library instead of a JPG library) this should actually be viewed as using a different application to render the object in the new format, even if the two rendering engines are "wrapped" as a one single, unchanged application. The potential side-effect is thus even bigger than when emulating the original rendering environment of a digital object. To judge if there are side-effects due to the change in the rendering environment we have to compare the new rendering to the original rendering, just as we would do in an emulation setting. As with an emulation action, any change in the new view-path has to be re-evaluated. We need to keep in mind that it is not enough to compare the migrated object in the current environment and assume that the rendering will be the same in a future environment. If the digital object is migrated while preserving it for the long-term it has to be validated against the original rendering of the object to be able to make a statement about the authenticity of the rendering.

As an example for the changes in object format and its influence on the rendering a document in Microsoft Word for Windows 97-2003 (doc) format was rendered using both Microsoft Office 2007 (Figure 4a) and OpenOffice 3.4 (Figure 4b), both running on a PC using Windows 7 operating system. While small differences in the rendering are visible (e.g. horizontal spacing), the properties of the document considered significant for most settings are unchanged. Next, the document has been migrated to Microsoft Word for Windows 2007 (docx) format. The document is rendered exactly identical in Microsoft Word 2007, so the significant properties stored in the file (layout information, content, metadata, etc.) are still present. However, in OpenOffice 3.4 the rendering looks completely different to a state that is unusable for most applications (Figure 4c).

Thus, any object with preservation actions applied, whether emulation or migration, must be evaluated in the context of its rendering environment, as



Standard Form 15 (Rev. 2/90) (EG)
U.S. Office ot Personnel Management
FPM Supplement 296-33
FPM Chapter 211
APPLICAT
VETER/
(TO BE USED BY VETER/
PERSON APPLYING FOR PREFERENCE
1. Name (Last, First, Middle)
2. Nan
3. Home Address (Street Number

Standard Form 15 (Rev. 2/90) (EG)
U.S. Office ot Personnel Management
FPM Supplement 296-33
FPM Chapter 211
APPLIC/
VETEI
(TO BE USED BY VETE
PERSON APPLYING FOR PREFERENCE
1. Name (Last, First, Middle)
2. N
3. Home Address (Street Number, (

Standard Form 15 (Rev. 2/90) (EG) U.S. Office ot Personnel Management FPM Supplement 296-33 FPM Chapter 211
APPLICATION FOR 10-POINT VETERAN PREFERENCE (TO BE USED BY VETERANS & RELATIVES OF VETERANS)
PERSON APPLYING FOR PREFERENCE

Fig. 4. Snippets of the same layout region of a document in MS Word for Windows 97-2003 format rendered in MS Office 2007 (a), OpenOffice 3.4 (b) and the same document migrated to MS Word for Windows 2007 (docx) format rendered in OpenOffice 3.4 (c)

changes to significant properties may occur that are not properly detectable from an analysis of the static object properties only.

7 Conclusions and Future Work

In this paper we showed how the generic view-path for the rendering of a digital object is composed. We took a look at the different levels on which emulation can be performed and how the digital preservation strategies migration and emulation on the different levels affect the view-path of a digital object. We then compared the effects of the two strategies and showed that the change in view-path makes it necessary to take the rendering environment into account when evaluating any digital preservation action, be it a migration or an emulation action.

Based on these observations we think that preservation planning strategies have to be adapted when using migration as a digital preservation strategy. To ensure an authentic rendering it is necessary to take the target rendering environment after migration into account. Any change in the target rendering environment (i.e., a layer in the view-path becoming obsolete) has to lead to a new preservation planning iteration evaluating the changed environment, even if the format of the digital object stays unchanged. This makes the evaluation process for migration strategies essentially identical to the evaluation of emulation strategies. So, after all, it seems that migrations and emulation digital preservation actions are rather birds of a feather, than two different kinds of beasts.

Acknowledgments. This research was co-funded by COMET K1, FFG - Austrian Research Promotion Agency and by the European Commission under the IST Programme of the 7th FP for RTD - Project ICT-269940/TIMBUS.

References

1. Becker, C., Kulovits, H., Guttenbrunner, M., Strodl, S., Rauber, A., Hofman, H.: Systematic planning for digital preservation: Evaluating potential strategies and building preservation plans. Int. Journal on Digital Libraries 10(4), 133–157 (2009)
2. Becker, C., Kulovits, H., Rauber, A., Hofman, H.: Plato: a service-oriented decision support system for preservation planning. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL 2008). ACM (June 2008)
3. Becker, C., Rauber, A., Heydegger, V., Schnasse, J., Thaller, M.: A generic XML language for characterising objects to support digital preservation. In: Proc. 23rd Annual ACM Symposium on Applied Computing (SAC 2008), vol. 1, pp. 402–406. ACM, Fortaleza (2008)
4. Brown, A.: Automatic format identification using PRONOM and DROID. Digital Preservation Technical Paper 1 (2008), http://www.nationalarchives.gov.uk/aboutapps/fileformat/pdf/automatic_format_identification.pdf
5. Guttenbrunner, M., Rauber, A.: A measurement framework for evaluating emulators for digital preservation. ACM Trans. on Information Systems 30(2) (2012)
6. Guttenbrunner, M., Wieners, J., Rauber, A., Thaller, M.: Same Same But Different – Comparing Rendering Environments for Interactive Digital Objects. In: Ioannides, M., Fellner, D., Georgopoulos, A., Hadjimitsis, D.G. (eds.) EuroMed 2010. LNCS, vol. 6436, pp. 140–152. Springer, Heidelberg (2010)
7. Rothenberg, J.: Preserving Authentic Digital Information. pp. 51–68. Council on Library and Information Resources Washington, D.C., USA (2000)
8. Slats, J.: Emulation: Context and current status. Tech. Rep. (2003), http://www.digitaleduurzaamheid.nl/bibliotheek/docs/white_paper_emulatie_EN.pdf
9. Thaller, M.: Interaction testing benchmark deliverable PC/2 - D6. Internal Deliverable, EU Project Planets (2008), http://planetarium.hki.uni-koeln.de/planets_cms/sites/default/files/PC2D15_CIM.pdf
10. van Diessen, R.J.: Preservation requirements in a deposit system. IBM/KB Long-Term Preservation Study Report Series Number 3, ch. 3 (2002), http://www-05.ibm.com/nl/dias/resource/preservation.pdf