Emulation as a Business Solution: the Emulation Framework

Bram Lohman
Tessella
President Kennedylaan 19
Den Haag, The Netherlands
bram.lohman@tessella.com

Bart Kiers
National Library of the Netherlands
Postbus 90407
Den Haag, The Netherlands
bart.kiers@kb.nl

David Michel
Tessella
26 The Quadrant
Abingdon, United Kingdom
david.michel@tessella.com

Jeffrey van der Hoeven
National Library of the Netherlands
Postbus 90407
Den Haag, The Netherlands
jeffrey.vanderhoeven@kb.nl

ABSTRACT

Emulation is often considered a technically very complex subject. The association with complexity has long prevented it from being considered in an end-to-end business solution for long-term preservation and access to digital collections.

The Emulation Framework solves this problem by automating the steps required to render an unknown digital object in its original environment: characterising the object to determine its type; determining the environment required to render that type of object; setting up the required software and emulators providing the hardware; and configuring the environment to properly render the digital object. Automating these steps allows a novice user to easily render a digital object in an environment for accessing it in its original form.

Each of the four steps of the emulation workflow are described in detail, providing a simple tool for managing a complex decision making process.

Keywords

Emulation, framework, digital preservation, workflow, business solution, KEEP, characterisation, technical environment, viewpath, software

1. INTRODUCTION

Long-term preservation of digital objects not only implies looking after their conservation, but also necessitates the development and execution of strategies to ensure these objects remain accessible and understandable in the future.

The KEEP [6] (Keeping Emulation Environments Portable) project is a research project co-funded by the European Union under the Seventh Framework Programme (FP7). It does research into media transfer, emulation and portability of software from a technical and legal perspective [15]. The project extends previous work on emulation, such as the Dioscuri project that developed an x86 emulator [1], and the Planets project which amongst others created emulation and migration services [8]. Emulation is a vital strategy for permanent access, but it requires several more steps to become mature [2]. KEEP aims to deliver a strategy that provides permanent access to multimedia content (such as computer applications and console games), not only now but also in the long term.

2. WHY EMULATION?

Emulation recreates a computer environment (target) on top of another computer environment (host) [11]. It is a proven technology that can be used to cope with obsolescence of hardware and software. By rendering a digital object within an environment running original software, an authentic recreation of that object in its native computer environment is given. The advantage of such a strategy is that no change to the digital object is required which offers better conditions for displaying it in its original form. Another advantage of emulation is that it also works for complex digital objects such as software applications (e.g. games), websites or visualisations of data sets.

3. AN END-TO-END EMULATION WORKFLOW

One of the main problems facing emulation is the lack of knowledge in identifying and configuring the technical environment required to render a digital object. The KEEP project recognises this issue, and in May 2011 released the Emulation Framework (EF), an open source solution for applying emulation as an access strategy for files and computer programs. It is released under the open source Apache 2.0 license and is freely available [3]. When a user requests an item from a digital collection and this item requires an ob-

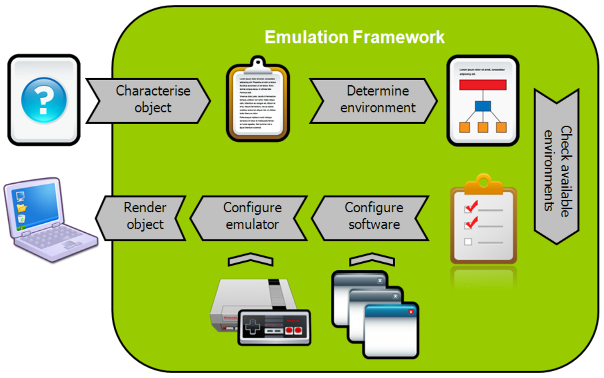


Figure 1: Emulation Framework workflow.

solete computer environment to render, the EF offers a solution that does not require any in-depth knowledge, following the workflow steps depicted in Figure 1.

The EF automates the identification of an (unknown) digital object; the need to know what application, operating system (OS) and hardware is required to emulate the object; preparing the selected environment for use; and configuration of the environment for rendering the digital object. These four steps are explained in more detail in the following sections.

3.1 Characterising an unknown digital object

Characterisation is a subject in digital preservation that has been researched in depth. This research has resulted in several tools that can characterise an unknown digital object, i.e. determine its file format. Harvard University Library Office for Information Systems released a tool, called the File Information Tool Set (FITS) [4], which acts as a wrapper for several proven open source tools. FITS identifies, validates, and extracts technical metadata for various file formats. It normalises, consolidates, and reports any errors in the output of the wrapped tools. FITS currently uses Jhove, National Library of New Zealand Metadata Extractor, DROID, FFIdent, Exiftool and File Utility [4]. It was an obvious choice to use this tool for characterisation in the EF.

The tools have no problem identifying the top 10 most common file types used in memory institutions [13]. Unfortunately, they lack support for most objects used in the emulation community: computer games, cartridges and disk image files created by that community. These disk images include, for example, common Amiga and Commodore 64 formats. During EF development, support for some of these formats was added by reconfiguring the FITS tool.

The FITS software also provides a novel selection method: it returns the number of tools that agree on the determined file format. This can be used as a measure of success, along with validation, and is used within the EF to automatically select the digital object's file format. Once the file format has been identified, the next step is to select an environment that provides the dependencies to render it in its original context.

3.2 Determining a rendering environment for a known digital object

The EF defines a rendering environment in a similar way as a viewpath [14] (or emulation pathway), of which two examples are shown in Figure 2. This is a structured description of the complete hardware and software stack needed to render a digital object. For today's PC architectures it consists of four layers (digital object, rendering application, OS, hardware platform), although for other architectures, not all layers are required. Console games, for example, usually only have two: digital object and hardware platform (including embedded software).

This is the simplest approximation of a rendering environment. Although it works for simple cases, in practice the connections between layers are more complicated: file formats require certain versions of applications to render properly; integration of application and OS requires specifics such as drivers and libraries; integrating OS and hardware platform depend on specific firmware to work together. In the current design, these more complicated cases are not supported and the EF uses the simple four-layer model of digital object file format, application, OS and platform.

Keeping the model simple does have an advantage. As complexity is increased exponentially at every layer – for each format there is often more than one application to support it; each application can run on different OS's, and each OS usually supports many hardware configurations (or emulators) – a model with fewer layers has lower complexity.

Technical metadata links each of the layers, and thus a directed graph can be generated with the digital object file format as the root node. The EF relies on an internal database containing metadata but can also interface with technical registries such as PRONOM [9] or PCR [16] to retrieve this metadata. To address the lack of publicly available registries, KEEP is addressing this issue [12] as well.

The EF is currently prototyping a novel method of combining information from different registries to ensure more robust information is used to create technical environments.

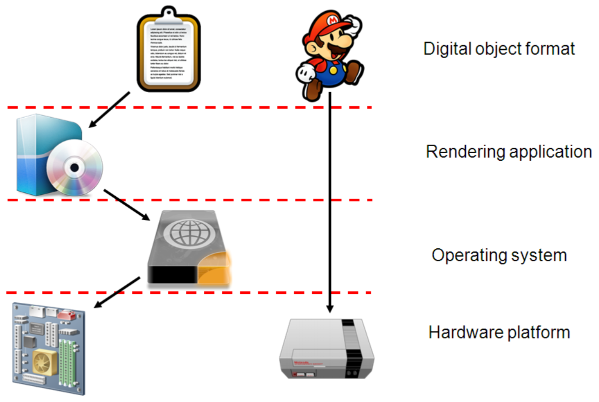


Figure 2: Environmental dependencies of digital objects.

3.3 Preparing the hardware and software stack

The main technical problem is merging the four distinct components defined by the digital object and its dependencies into one assimilated environment: to view a digital object, the bit stream has to be interpreted by an application, which in turn has to be configured specifically (i.e. installed) on an OS that is configured for a particular piece of hardware. At rendering time, the stack is difficult to view as individual components.

There are several approaches to generate an environment:

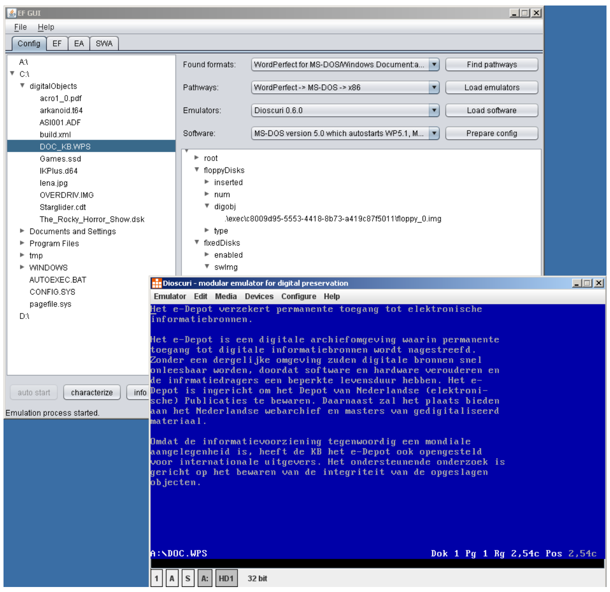
- Use an automated method to merge the four components at runtime.
- Prepare a complete environment beforehand to be used.
- Use a combination of these.

Although for simpler environments, such as MS-DOS, the 'merging' step can relatively easily be automated, software and hardware systems have in recent years become increasingly complex. Environments running on today's desktop are based on customised hardware running a specifically set up OS that has applications that are tightly coupled to it (e.g. registry entries, library versions, etc.). Setting up such an environment requires a high number of complex choices to be made. The problem is not so much that it can not be done, but there are so many corner cases and exceptions, that the effort of creating an automated method that can reliable generate any selection of environments far exceeds the benefits. The University of Freiburg is continuing research into this area [17].

The second approach requires setting up the OS, application and digital object as required by the selected environment in advance. The only reliable way is a human initiated, time-consuming process, but it only needs to be done once for an environment; it can then be stored and accessed whenever required.

The EF has tested the third approach as a proof of concept. In general, the digital object (top layer) and the hardware platform (bottom layer), are only loosely coupled to the layers in between. Those layers, the application and OS, however, are so interdependent, that only by setting these up beforehand can be guaranteed that it is done correctly. To address this, the EF created a 'Software Archive', a web service that holds prepared application/OS disk images along with metadata containing details of the OS, applications and hardware requirements. Using technical registry metadata, an appropriate disk image can be selected from the database that fulfils the environmental requirements.

Similarly, a separate web service, the 'Emulator Archive', holds the emulators that can be used to represent the hardware. It also contains metadata to match the required hardware selected in the technical environment, along with the type of software image it supports, and thus a match can be made between the emulated hardware and the OS/application layer.


EF GUI
File Help
Config EF EA SWA
Find pathways
Load emulators
Load software
Prepare config
auto start characterize info
Dioscuri - modular emulator for digital preservation
Emulator Edit Media Devices Configure Help
1 A S A: HD1 32 bit

```
A:\
C:\
  digitalObjects
    acro1_0.pdf
    arkanoid.t64
    ASI001.ADF
    build.xml
    DOC_KB.WPS
    Games.ssd
    IKPlus.d64
    lena.jpg
    OVERDRIV.IMG
    Starglider.cdt
    The_Rocky_Horror_Show.dsk
  Documents and Settings
  Program Files
  tmp
  WINDOWS
  AUTOEXEC.BAT
  CONFIG.SYS
  pagefile.sys
D:\

Found formats: WordPerfect for MS-DOS/Windows Document:a...
Pathways:      WordPerfect -> MS-DOS -> x86
Emulators:     Dioscuri 0.6.0
Software:      MS-DOS version 5.0 which autostarts WP5.1, M...

root
floppyDisks
  inserted
  num
  digobj
    .\exec\c8009d95-5553-4418-8b73-a419c87f5011\floppy_0.img
  type
fixedDisks
  enabled
  swImg

Emulation process started.
```

```
Het e-Depot verzekert permanente toegang tot elektronische
informatiebronnen.

Het e-Depot is een digitale archiefomgeving waarin permanente
toegang tot digitale informatiebronnen wordt nagestreefd.
Zonder een dergelijke omgeving zuden digitale bronnen snel
onleesbaar worden, doordat software en hardware verouderen en
de infrmatiedragers een beperkte levensduur hebben. Het e-
Depot is ingericht om het Depot van Nederlandse (elektroni-
sche) Publicaties te bewaren. Daarnaast zal het plaats bieden
aan het Nederlandse webarchief en masters van gedigitaliseerd
materiaal.

Omdat de informatievoorziening tegenwoordig een mondiale
aangelegenheid is, heeft de KB het e-Depot ook opengesteld
voor internationale uitgevers. Het ondersteunende onderzoek is
gericht op het bewaren van de integriteit van de opgeslagen
objecten.

A:\DOC.WPS                                Dok 1 Pg 1 Rg 2,54c Pos 2,54c
```

Figure 3: The Emulation Framework rendering a digital object in its original environment.

3.4 Configuring the environment to render the digital object

Configuration of a hardware platform, despite there being many different variations, can be made much simpler by creating a high-level hardware component model of it. Making the model sufficiently generic allows it to be used for multiple emulators. Although the low-level details for each hardware set may be different for each emulator (even if they address the same platform!), using a single model greatly simplifies the problem. The EF currently manages to configure 7 emulators covering 6 platforms using a single abstracted model.

To generate an emulator-specific configuration, the EF makes use of a template processor, a software component designed to combine a data model with a template to produce a result document [5]. Each emulator specific template contains the grammar for configuring that emulator, which when combined with the emulator-agnostic data model, generates the emulator specific hardware configuration that the software requires. Given the configuration options for the environment (such as number and type of floppy drives, hard disk parameters, CPU settings, etc.) a customised configuration can be created for each emulator.

The last part of this step is providing the digital object to the application within the disk image. Because the application and OS disk image is prepared prior to the process, the digital object cannot easily be inserted into it. However, it can be attached to the emulated hardware as a separate disk image that with the right configuration can allow the application within the disk image to access the digital object. For example, a disk image containing MS-DOS and WordPerfect is provided to an x86 emulator as a hard-disk, and a floppy disk image containing the accompanying WordPerfect file is also provided to the platform. Within the rendering environment, it is possible to boot the OS from the hard-disk, start the application, and read the digital object from the attached floppy disk from within the application.

This completes the last step of the workflow to render an unknown digital object in its original environment, as can be seen in Figure 3.

4. BUSINESS INCENTIVES

With the release of the EF using emulation tools has become more accessible, bypassing difficult setups or technical restrictions. The EF runs on Java, making it compatible with all mainstream computer platforms. Moreover, management of required emulators and software packages has become more organised by using the service-oriented Emulator and Software Archives. With the large number of freely available emulators, most computer platforms can be emulated by including them in the EF. However, care must be taken when using old applications and emulators as software licenses and hardware patents can restrict limitations of use [15]. For this reason the current release of the EF only uses open source emulators and applications.

5. ONGOING RESEARCH

Building on the first release of the EF (May 2011), the KEEP project is working on improving the software. Two new releases are planned before the end of the KEEP project in February 2012. These will incorporate the user feedback from tests performed by the Bibliothèque nationale de France, Dutch National Archives, Computerspiele Museum Berlin, research institute CERN and the Netherlands Media and Art Institute. Altogether these organisations represent five major domains: library & archiving, culture, research and art.

Furthermore, KEEP is doing research into remote emulation, with the goal of accessing the rendered digital environment from a thin client. This will move the high requirements emulators place on the underlying hardware to a server specifically built for the task.

6. CONCLUSION AND BENEFITS

The EF has shown that an end-to-end business solution using emulation to render a digital object in its original environment is feasible. The EF is currently freely available, allowing individuals and institutions to take advantage of the possibilities to unlock their digital archive to the wider public at a very low total cost of ownership. Especially those digital objects for which migration, currently the main digital preservation strategy, provides no accessibility is this solution an alternative.

Ongoing pilots at the National Library of the Netherlands, the German Computerspielemuseum and integration with Tessella's Safety Deposit Box [10] show that in a wide range of environments the EF offers access to a large set of digital objects. With the Open Planets Foundation [7] in place, a platform exists that will ensure this solution continues to be developed, and also guarantees continual support.

All in all, the EF offers an effective way of ensuring long-term access to practically any digital object, and can be put to use by any user or institution regardless of the technical knowledge available.

7. ACKNOWLEDGMENTS

KEEP has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no ICT-231954.

8. REFERENCES

[1] Dioscuri — the modular emulator. `http://dioscuri.sourceforge.net/`. Accessed: 01-Sep-2011.

[2] Emulation Expert Meeting Statement. `http://www.kb.nl/hrd/dd/dd_projecten/projecten_emulatie-eemstatement-en.html`. Accessed: 01-Sep-2011.

[3] Emulation Framework. `http://emuframework.sourceforge.net`. Accessed: 01-Sep-2011.

[4] File Information Tool Set (FITS). `http://code.google.com/p/fits`. Accessed: 01-Sep-2011.

[5] FreeMarker — Java Template Engine Library. `http://freemarker.sourceforge.net`. Accessed: 01-Sep-2011.

[6] KEEP project. `http://www.keep-project.eu/`. Accessed: 01-Sep-2011.

[7] Open Planets Foundation. `http://www.openplanetsfoundation.org`. Accessed: 01-Sep-2011.

[8] Planets project. `http://www.planets-project.eu/`. Accessed: 01-Sep-2011.

[9] PRONOM — the online registry of technical information. `http://www.nationalarchives.gov.uk/PRONOM`. Accessed: 01-Sep-2011.

[10] Safety Deposit Box. `http://www.digital-preservation.com/solution/safety-deposit-box`. Accessed: 01-Sep-2011.

[11] What is Emulation? `http://www.kb.nl/hrd/dd/dd_projecten/projecten_emulatiewatis-en.html`. Accessed: 01-Sep-2011.

[12] D. Anderson, J. Delve, D. Pinchbeck, L. Konstantelos, A. Lange, and W. Bergmeyer. D3.3 final document analyzing and summarizing metadata standards and issues across Europe. Technical report, KEEP project, September 2010.

[13] S. v. Bussel and F. Houtman. Gap analysis: a survey of PA tool provision. Technical report, Planets project.

[14] R. Diessen and J. Steenbergen. Long Term Preservation Study of the DNEP Project. Technical report, IBM, National Library of the Netherlands, December 2002.

[15] J. v. d. Hoeven, S. Sepetjan, and M. Dindorf. Legal Aspects Of Emulation. *iPRES 2010 proceedings*, July 2010.

[16] L. Montague and S. v. Bussel. PLANETS Core Registry: Future Vision Document. Technical report, The National Archives, National Library of the Netherlands, May 2010. PLANETS project, PC3-D24.

[17] D. v. Suchodoletz, K. Rechert, J. Schroder, and J. v. d. Hoeven. Seven steps for reliable emulation strategies solved problems and open issues. *iPRES 2010 proceedings*, July 2010.