Towards a Universal Virtual Interactor (UVI) for Digital Objects

Euan Cochrane
Yale University Library
United States of America
euan.cochrane@yale.edu
0000-0001-9772-9743

Seth Anderson
Yale University Library
United States of America
seth.r.anderson@yale.edu
0000-0003-3304-8484

Ethan Gates
Yale University Library
United States of America
ethan.gates@yale.edu
0000-0002-9473-1394

Klaus Rechert
OpenSLX GmbH
Germany
klaus@openslx.com
0000-0002-8807-018X

Jessica Meyerson
Educopia Institute
United States of America
jessica@educopia.org
0000-0002-0180-9616

Abstract – Practitioners generally agree that providing a service to enable opening and interacting with born digital objects in their "original" software is valuable for historians, researchers and the general public so that they can experience full-fidelity experiences of the objects. Enabling this has, to date, been a difficult, time-consuming, relatively resource intensive, and tedious. In this paper, we show how we are on the verge of creating a new method and series of tools to simplify and automate the process of interacting with digital objects in their original software and greatly reduce the time and resource costs of doing so. We outline the history of the developments in the areas of emulation and software preservation that we have built on and we outline the concept of this set of tools and processes we call the "Universal Virtual Interactor". We also discuss how the UVI is being created, and finally we discuss how it may be improved upon in the future and how it may be implemented in access and discovery tools.

Keywords – Emulation, Access, Rendering, Interactivity

Conference Topics – 2. Designing and Delivering Sustainable Digital Preservation; 5. The Cutting Edge: Technical Infrastructure and Implementation.

I. Introduction and Background

From at least the 1980s, many years prior to the publication of Jeff Rothenberg's seminal article and paper "Ensuring the Longevity of Digital Documents" [1] in 1995 there have been advocates amongst digital preservation practitioners for preserving software as both information in itself but also, and importantly, as a utility for accessing other/existing digital objects over the long term [2]. As Rothenberg said:

"they [future generations] should be able to generate an emulator to run the original software that will display my document."[1 p47]

As the National Library of the Netherlands so eloquently articulated in 2003:

"There is a difference between paper and digital records. Any paper record can be perceived through the five human senses; no digital record can be perceived without going through computer hardware and software..... .

....Digital records are software dependant. They rely upon the software that was originally intended to interpret (or display) them. When that software becomes obsolete, perhaps within the space of a few years, the problem arises of how to read that record without its original software application. It is unlikely that different versions of the application will read the file in the same way, and this may well result in a change in the interpreted record (the visible or available view of the file) that affects its archival integrity. Some data may be lost altogether; in other areas, data may be gained. There may be no way to compare a new version with the original, so changes may go unnoticed. Any changes to the

16th International Conference on Digital Preservation
iPRES 2019, Amsterdam, The Netherlands.

record may affect its authenticity and integrity, which in turn may affect its archival and legal status. Depending on the nature of the record and its use, this can cause problems, not least that of losing or misrepresenting history."[3]

This was further illustrated in the "Rendering Matters" [4] research undertaken in 2011 at Archives New Zealand which demonstrated (with visual examples [5]) the necessity of ensuring we can interact with preserved digital objects using the original or representative contemporaneous software environments.

While there continue to be many valiant efforts to preserve computing hardware for future generations, particularly for pedagogical purposes[1] this approach is unfortunately neither economically scalable nor likely to be sustainable over long time frames [6]. For this reason, practitioners over the last 20+ years have instead focused on preserving the software component(s) and ensuring we can continue to maintain the ability to run legacy software as the hardware that supports it has become obsolete.

In Jeff Rothenberg's 1995 article [1], and his subsequent work with the National Library and the National Archives of the Netherlands, Rothenberg argued that emulation was likely to be the only effective general strategy for preserving the complete full-fidelity experience of digital objects over time. Rothenberg has also argued that emulation is cost-effective as a just-in-time rather than a just-in-case approach.

"few organizations can justify the cost of translating documents that they no longer use." [7 *p13]*

Emulation also couples well with other long-term digital preservation tools and strategies, such as normalization and migration, with the former facilitating cost-effective preservation of fidelity and authenticity (and "digital patina" [8]), and the latter facilitating reuse of components of digital objects that can be easily extracted (potentially on-demand) from their native contexts. Emulation can also become a tool for performing just-in-time migration when coupled with macros that interact with emulated software environments to run "open-file-then-save-as-a-new-format" operations [9].

A fruitful way of interpreting the history of emulation tools in digital preservation is to consider it as an attempt to maximise the preservation impact of our preservation tools while minimizing long-term support costs. To this end, in 2001 Raymond Lorie, while working at IBM, developed the initial design concept for what he called a "Universal Virtual Computer":

"We propose to save a program P that can extract the data from the bit stream and return it to the caller in an understandable way, so that it may be transferred to a new system. The proposal includes a way to specify such a program, based on a Universal Virtual Computer (UVC). To be understandable, the data is returned with additional information, according to the metadata (which is also archived with the data).

...we propose to describe the methods as programs written in the machine language of a ***Universal Virtual Computer (UVC)****. The UVC is a Computer in its functionality; it is Virtual because it will never have to be built physically; it is Universal because its definition is so basic that it will endure forever. The UVC program is completely independent of the architecture of the computer on which it runs. It is simply interpreted by a UVC Interpreter. A UVC Interpreter can be written for any target machine."* [10]

The UVC had quite a few pitfalls, primarily that new UVC code had to be written for every new file format. But the general approach, that of stabilizing and preserving the functionality at the "highest" possible level in the interpretation set in order to minimise the number of code revisions required to keep the overall functionality operating, was sound.

Building on this the Koninklijke Bibliotheek, National Library of the Netherlands (the KB) developed "Dioscuri" a "modular emulator" that could run anywhere the Java Virtual Machine (JVM) could run [11].

[1] Such as the work of the Living Computers Museum + Labs (https://livingcomputers.org/), Computer History Museum (https://www.computerhistory.org/), Media Archaeology Lab at University of Colorado Boulder (https://mediaarchaeologylab.com/), Maryland Institute for Technology in the Humanities (https://mith.umd.edu/0), retroTECH at GeorgiaTech University (http://retrotech.library.gatech.edu/)

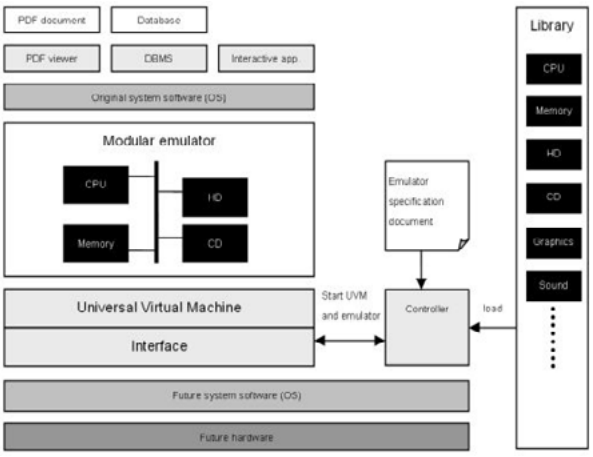

Figure 1: The Dioscuri Modular Emulator Design

Dioscuri was intended to be extensible over time with new modules being created for newer systems as needed. In the Dioscuri model the component that was to be stabilized and preserved was the Java Virtual Machine, i.e. future generations would only have to re-enable write the JVM code in the new computing system in order to maintain access to all the modular emulators and the systems and software that they support.

Following from and incorporating the Dioscuri work, the Keeping Emulation Environments Portable (KEEP) project [12] developed a desktop application that bundled a set of emulators with a configuration GUI that incorporated the concepts of Preservation Layer Models and "View Paths" from early IBM/KB work [13]:

"The PLM outlines how a file format or collection of similar objects depends on its environment. A PLM consists of one or more layers of which each layer represents a specific dependency. The most common PLM consists of three layers: application layer, operating system layer and hardware layer. However, other variations can also be created. Based on a PLM, different software and hardware combinations can be created. Each such combination is called a view path. In other words, a view path is a virtual line of action starting from the file format of a digital object and linking this information to a description of required software and hardware" [14 p148].

To implement this the KEEP developers included a method for associating file formats with configured emulated environments such that you could submit a file and it would "automatically" be attached to an environment and that environment be loaded on your desktop to be interacted with. Overall the approach of the KEEP project was less efficient than that outlined by the UVC or Dioscuri alone as it incorporated multiple emulators that would have to be supported over time. However, it did introduce the dramatic efficiency of being able to use existing file-interpreters (E.g. commercial software applications) and the ability to reuse off-the-shelf emulators developed by third parties. Concepts and approaches that are included in the contemporary work we discuss further below.

At the same time as the KEEP, the Planets Project [15] had a sub-project to build the Global Remote Access to Emulation Services (GRATE) service [14]. A method for remotely accessing emulated environments via a web browser. This approach enables resource-intensive emulation to be managed and executed remotely while the user interacts with it through a browser-based viewer.

The GRATE project was led by a team at the University of Freiburg and evolved into what became the Baden-Württemberg Functional Long-Term Archiving (bwFLA) project [16]. The bwFLA project, in turn, developed the suite of tools now commonly referred to as Emulation as a Service or "EaaS". The EaaS tools follow the basic approach pioneered with KEEP, but implemented with a browser-based interface, while adding features such as enabling the definition of derivative disk images (more on this below), the separation of objects, environments and emulators, and the addition of many reliability improvements. The browser-based approach is transformative from a user-perspective as it democratizes access. All one needs to interact with an emulated computer is a web interface.

It is upon the EaaS infrastructure that we are building the EaaSI program of work. We're expanding on the concepts of a PLM and view path to create what we are calling a Universal Virtual Interactor. The goal of the UVI project is to develop a framework into which organizations and consortia can add legacy software and metadata in order to automate presenting digital objects to users for interaction in a web browser. The objects are presented in "original" or representative interactive computing environments utilizing original or representative software

16th International Conference on Digital Preservation
iPRES 2019, Amsterdam, The Netherlands.

from a time period that is appropriate to the object. Or, more succinctly, the UVI automates opening old files in their "original" software in a web browser.

II. Configuring Environments in the Emulation as a Service Infrastructure (EaaSI) program of work

A. *The EaaSI Program of Work*

In the Emulation as a Service Infrastructure (EaaSI) program of work we are working with partner organizations who are hosting EaaSI nodes running instances of Emulation as a Service: the "EaaSI Network [17]". Together with a local team at Yale University Library we are configuring and documenting emulated computing environments and enabling the environments to be shared between nodes in the EaaSI network. Upon that base we are building services, workflow interfaces, and APIs to perform various digital preservation and curation functions. One of these services is the UVI that relies on this set of configured computing environments for its core functionality.

B. *Hardware Research and Configuration*

The configuration of computing environments within the EaaSI program is a fairly involved process that is time consuming and deliberately thorough. We have a team recruited from students at Yale University who are performing this important role. The workers in the environment configuration and documentation team start by selecting an application and check to see what hardware and operating system it requires. Assuming the required environment doesn't exist, they next create a virtual hard disk that is stored as an image on our servers. The workers then configure an emulated computer that has the hardware specifications required to run the dependent operating system and also to run the application itself.

We do our best to match the emulated hardware specification to representative hardware from the period during which the software was most popular, or the period we are targeting to emulate. For example, for a Windows 98 computer we can choose to emulate a contemporaneous CPU (e.g. Pentium 3), volume of RAM (e.g. 256 megabytes), and compatible sound (e.g. a SoundBlaster 16), video (e.g. a Cirrus CLGD 5446 PCI) and network cards (e.g. an AMD PCNet PCI). Sometimes this requires historical research and we have consulted various online resources from old advertisements to compiled lists of hard drive prices over time.

Figure 2. An advertisement for Cybermax Personal Computers from the late 1990s, via user @foone on twitter

Historical and performative accuracy is also weighed against long-term costs. We aim to minimize the hardware variants that we support in order to reduce the long-term cost of moving the environments to new or migrated emulators.

C. *Documentation Operating System Configuration*

The configured computer is then documented as structured metadata and defined as a configured "hardware environment". These "hardware environment" combinations can be saved as templates in our system[1], allowing future users to reuse that

[1] The hardware environments are also matched to the software applications that we later install on the hardware environments and confirm their compatibility with. These applications also have their published hardware requirements documented and associated with them. In the future we hope to use these two sets of data to automate matching newly added software applications to pre-configured "compatible" hardware environments by matching the published hardware requirements of the new software with pre-configured environments that we have confirmed are compatible with the same requirements set.

16th International Conference on Digital Preservation
iPRES 2019, Amsterdam, The Netherlands.

configuration when selecting their requirements (either automatically or manually using a GUI) without having to configure every sub-component (for example, they might just select the most popular pre-configured hardware environment template that supports Windows 98 SE). Next a disk (image) is connected to that emulated computer, the operating system installation media is also attached, and the computer is switched on ("booted"). The configuration user can then run through the operating system installation and configuration. Throughout this process the configuration user has to make a number of decisions about operating system configuration and settings. These decisions can affect the functions of the operating system and the applications that come bundled with it, or may be run on it, in the future. For example, setting the resolution of the desktop will affect how software displays, or choosing a language or set of locale settings can dramatically change the user experience. EaaSI configuration users select from menus and pick lists to document each of the decisions they make and add new metadata options to those pick lists where necessary. This ensures consistency and machine-readability of the captured documentation/metadata.

Having configured and documented the operating system the configuration user shuts down the emulated computer and saves the results into the disk image file. This disk image and its documented contents are defined as a new "software environment"[1]. This software environment is documented by the configuration user as structured metadata and assigned a unique identifier. Together with the hardware environment they are defined as a "computing environment" which is also documented and assigned a unique identifier. We use these concepts to organize and enable discovery of assets within the EaaSI interface.

[1] An important tangential benefit of this approach is that by preserving just one of these environments, such as a Microsoft Windows 98 computing environment running Microsoft Office 97, we have ensured that the very many digital objects created by and made accessible using the applications in the Office suite are able to be accessed for future generations. Once we have one of these environments configured we can reuse it to re-enable interaction with all of those countless digital objects at minimal incremental cost and on an on-demand basis that is useful from a financial planning perspective as it matches the burden of cost to the time of access.

D. Installing and Documenting the Application Software

Our next step is to install the selected application onto the existing software environment to create a new software environment. Fortunately, the EaaS software facilitates minimising the incremental cost and associated environmental impact of this by enabling the creation of "derivative" disk images that are "derived from" an existing image (either a full disk image or a derivative itself) [87]. The changes that a configuration user makes when installing and configuring the added application are all that is captured onto disk in the resultant derivative file. When the associated new software environment needs to be used in the future the full disk image (or hierarchy of image and derivatives) and the derivative file are brought together at the time of execution and integrated in real-time by the EaaS software.

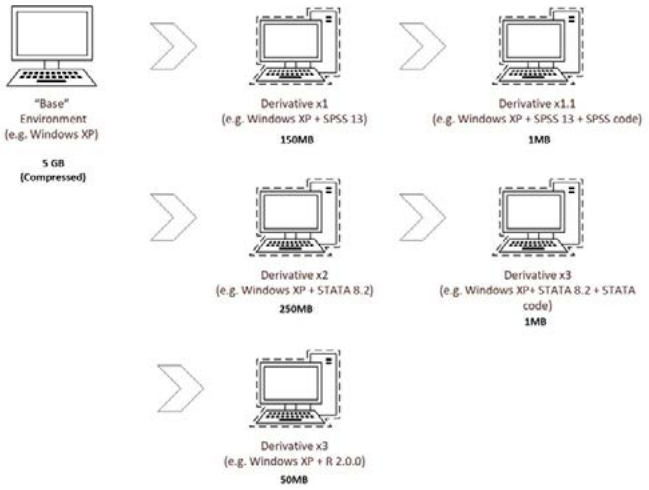

Figure 3. An illustration of the storage cost savings enabled by the use of derivative disk images

This greatly reduces the storage burden of preserving multiple software environments and frees configuration users to pre-configure software environments with only minor differences between them without raising significant concerns about storage costs. Given that it can take a number of minutes to load a computing environment and make even a small (but potentially very useful) settings change, the benefit of this becomes clear: by pre-configuring multiple environments just once each and then sharing them, this greatly reduces the time required for future users to provision a software environment appropriate to their use case, i.e. users can just pick the pre-configured software environment they want from a list.

DOI: 10.1145/nnnnnnn.nnnnnnn

While installing and configuring the application in the software environment the configuration user documents a number of facts about it in order to facilitate automated interaction with the environment in the future. For example, the configuration user will document every relevant executable program included in the application and various facts about it such as:

1. Where the executable is located within the file system
2. How the executable itself can be initiated programmatically (at system start up)
3. How the executable can be made to open a digital object programmatically during the initiation process
4. What file formats the application can open

a. This includes documenting the exact description of the format and its extension (where applicable) as displayed in the application's user interface[1]

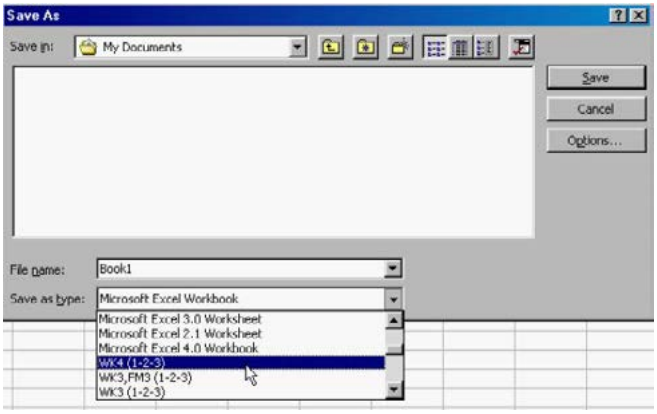

Figure 4: Save-as types as presented in Microsoft Excel 97

5. What file formats the application can import

• As above - including the specific displayed parameter.

6. What file formats the application can save to

• As above - including the specific displayed parameter.

7. What file formats the application can export

• As above - including the specific displayed parameter.

The configuration user also makes an attempt to research when the application was first released, when it was most popular, when it was first superseded (and by what software) and when it went out of general use or was no longer supported. And finally, they document the default mime-types or file extensions (depending on the operating system) that the software is configured to automatically open at the operating system level within the software environment (i.e. which types of files the application will automatically open when the file is double clicked within the operating system).

The configuration user then shuts down the emulated computer and saves a derivative environment which gets added to the pool of available software environments. Where possible the configuration user will then publish the environment so that others participating in the EaaSI network can add the environment to their local pool.

The software behind the EaaSI network is open source and available on GitLab [19], and while the current EaaSI network is using the fair-use rights available under copyright law in the United States of America to facilitate the sharing of environments outlined here [20] there is no technological reason the software and approach couldn't be extended internationally. The Software Preservation Network [21], an international cooperative of stakeholders in software preservation, is working with international partners to explore avenues for expanding the EaaSI network beyond the United States and/or to enable similar networks to be established in other legal jurisdictions.

III. The Universal Virtual Interactor (UVI)

The Universal Virtual Interactor or UVI is a concept built on the history outlined above. Its name is descriptive of its functionality: it is intended to be Universal and apply to all digital objects. It is Virtual as it uses emulation and/or virtualization ("Virtual" is also included in homage to the UVC concept described above). And it, like the objects it is used with, is Interactive.

The UVI is intended to automatically present a digital object for interaction by a user in a software environment that is either the original that the object was created and used within, or a software

[1] Such detailed application-specific metadata may be useful for distinguishing between functionally different file format variants such as "Excel 4.0 created by Microsoft Excel 97" vs. "Excel 4.0 as created by Quattro Pro 8".

environment that represents one that would have been in use at the time the object was created (and/or soon after). To enable this functionality the UVI attempts to automatically map attributes of digital objects to pre-configured software environments that could be used to interact with them. It dynamically generates view-paths based on analysis of the digital objects that are submitted to it and the metadata available about the configured environments that exist within the EaaSI network. For example, the UVI might analyse a ".doc" file and identify that it was created with WordPerfect 5.1 for MS-DOS and automatically match it to an available emulated environment containing that software. Similarly, it might identify that a ".xls" file was created in OpenOffice Calc 1.0 and match the file to a pre-configured emulated environment containing that software, as it is most likely to be the "best" option to use for interacting with that file (despite .xls being the default format for Microsoft Excel and not OpenOffice Calc 1.0).

The UVI's algorithm analyses a number of factors to identify these view-paths including:

1. It analyses the dates associated with files in the digital object (e.g. last edited and earliest created) and the available file format information and attempts to identify which environments can be used to interact with the object. The dates may come from file system metadata or embedded metadata and are evaluated for trustworthiness using a variety of tests.
2. It attempts to match the data ranges within the digital objects to first identify what software was in use and popular at the time the object was created and in use.
3. It then identifies, of that set, which of the applications could open or import the objects.
4. In additional steps, it then uses further metadata (where available) to attempt to further reduce the list of possible interaction environments, details such as which applications were popular at the time, which application created the object (information that may be inferred from metadata within the file and from information about applications that were available contemporaneously with the file), or which application created those files by default and was also generally used to interact with them.

The algorithm evolves in response to additional configuration metadata contributed by EaaSI Network users. The result is a list of environments that are available in the EaaSI network with weightings associated with how likely they are to be an appropriate representative (representing an environment that would have been used at the time the object was in use) environment to interact with the object.

Developers of discovery and access systems can choose how they want to use that list of environments. They may choose to present all options to a user, only the highest weighted option, randomly assign an environment, or use some other approach. The UVI is agnostic about this decision.

Once the files have been mapped to environments the system has a number of options for ensuring the objects are made interactable to users. In all cases the object is made available to the environment either by:

1. Including the object in a disk image that is attached to an environment
2. Editing the disk image to insert the object into a location in the file system.

We then have multiple options for enabling the content to be opened in the target software application within the emulated software environment.

1. The environment can be presented to a user with text instructions indicating how to open the object using the interface of the emulated environment
2. The disk image of the core operating system can be edited to force the object to execute on start-up (e.g. by placing the file or a link to it in the "startup" folder of a Windows environment or by inserting a script into a Linux boot process that utilizes the executable syntax metadata referenced above to open the digital object with a particular executable upon system start-up.
3. A mouse/keyboard input macro can be run after the system has loaded in order to open the object in the appropriate software.

The end result of all of these configurable options is that after a user clicks on a link to an object in a web browser they are quickly presented with the ability to interact with it in an "original" or representative software environment from the era of the object.

A. Progress

The EaaS and EaaSI teams already have some of the components available to enable this automation and are rapidly building and creating more. The basic tooling to automate the steps of connecting an object to an available software environment and editing the disk image to make the object execute at system boot time are already available. A limited version of this approach is used in the German National Library to automatically present CD-ROMs to users in emulated computers running operating systems that the CDs should be compatible with [22].

The EaaSI team have acquired a large software collection and have begun configuring and documenting software environments using the approach outlined above. The initial prospects are promising and we're aiming to have 3000 software environments configured by July 2020.

IV. Future work

Programmatic Interaction with Environments

Early work with migration by emulation was completed as part of the PLANETS project by the partners at the University of Freiburg. This approach involved automating attaching a file to a computer environment, within a disk image, loading the environment, then running a macro/program that uses pre-recorded and automated mouse and keyboard inputs to open the file in an application and save the contents into a new file with a different format. This tooling still exists behind the scenes in the Emulation as a Service software that is a direct descendant from the work of the PLANETS project and a core part of the UVI. Using these features the environments created for the UVI could be re-purposed not just to serve as tools for "manual" interaction but also to be used as tools for automated interaction with at least two potential use cases:

1. The aforementioned migration by emulation, including daisy-chaining migration steps using multiple software environments.
2. Enabling "distant reading" [23] of a variety of different software environments or of sets of diverse digital objects using the same software environment.

a. For example, a researcher may be interested in comparing changes in user interfaces over time by automatically loading, automatically interacting with, and analysing the output of the environments over time. Or a researcher may be interested in automatically comparing the rendering of one digital object in a diverse variety of different software environments by automatically opening the same object in a variety of different software environments, interacting with them automatically, and analysing the outputs.

We are also following the work of the Preservation Action Registry (PAR) project [24] with great interest. As we develop persistent identifiers for computing environments there is the potential to incorporate emulation view paths into PAR with the UVI as the "tool" involved. As discussed in [25] this would enable digital preservation system developers to match digital objects to UVI compatible software environments during the ingest process and to use this information to enable access tools to automatically present the object in the appropriate environment when it is requested for access. Additionally the migration pathways enabled by the migration-by-emulation functionality should expand the PAR dataset extensively.

Our UVI is machine/algorithm driven and so the more environments that are available and the greater diversity between them, the more powerful the UVI becomes. However, we don't yet have Artificial Intelligence (AI) algorithms available to do the kind of configuration and documentation tasks described in section C. above. We are currently manually pre-configuring multiple slightly different environments and describing them with machine readable metadata. For example, we configure the same environment with multiple different pre-configured display resolutions to enable users/machines to just pick a pre-configured option rather than having to make the configuration change themselves. In the future we would like to explore using programmatic interaction with the environments to both configure and document new environments in order to further reduce the cost of populating the EaaSI network and improve the effectiveness of the UVI.

16th International Conference on Digital Preservation
iPRES 2019, Amsterdam, The Netherlands.

B. *Integration into Discovery and Access Systems*

The UVI is being built as a set of APIs that enable developers to either:

1. Request an object opened in a specific environment and get back the information required to embed the environment for interaction in a browser window.
2. Submit an object and request a list of potentially appropriate environments for use in interacting with the object, with weighting data to aid in selection/presentation to a user
3. Submit metadata (dates and file format information) and request a list of potentially appropriate environments for use in interacting with the object, with weighting data to aid in selection/presentation to a users
4. Submit a file or metadata and receive back the information required to embed the likeliest appropriate environment in a web browser.

This flexibility provides developers with a number of options for how they integrate the UVI into their discovery and access workflows. They may wish to provide more or less options to end-users and may already know which environments they want to use for particular digital objects.

As discussed above, a version of this approach is already in use in the German National Library [22] and the EaaSI nodes are aiming to explore integrating the UVI into their access and discovery systems beginning in 2020.

C. *Reducing time to load environments*

The EaaS team at the University of Freiberg and their commercial offshoot OpenSLX GmbH have been working to enable computing environments to be paused at a point in time and restarted instantaneously. That functionality coupled with macro-based interaction with environments would enable reducing the time from clicking on a digital object in a finding aid or catalogue and having it presented to you in your web browser. An environment could be instantaneously loaded with an object attached in a disk image and a macro immediately run that opens the file using keyboard/mouse interactions. This could also be managed such that the user doesn't get presented with the environment in their browser until the macro has been completed ensuring no conflicts between the macro-driven inputs and the user's manual inputs.

V. Conclusion

The UVI is the conceptual legacy of more than two decades of applied research on emulation in cultural heritage contexts including the Planets Project, the UVC project and Jeff Rothenberg's early work and research with the Dutch National Library and Archives. Our current work on/within the EaaSI program further reduces barriers to using emulation and preserved software as a means of interacting with preserved digital objects. While detail-heavy and time-intensive, the collective efforts of the EaaSI Network will pay dividends in the future through economies of scale. When the environments and tooling we are developing re-enable access to potentially limitless digital objects that might otherwise be inaccessible or lose significant fidelity and content, their value will be clear. Additionally, once the UVI is standardised we will have the opportunity to open up additional services and integration points to spread the benefits throughout the digital preservation community and on to the public at large.

16th International Conference on Digital Preservation
iPRES 2019, Amsterdam, The Netherlands.

References

[1] J. Rothenberg, "Ensuring the Longevity of Digital Documents", Scientific American, Vol. 272, Number 1, pp. 42-7. 1995.

[2] J. Meyerson, "Software Preservation Literature Review: The Co-Determinacy of User Needs and Advances in Preservation Methods", 2014,https://www.softwarepreservationnetwork.org/blog/software-preservation-literature-review-2014/, accessed 03/19/2019

[3] J. Slats Et al, "Digital Preservation Testbed White Paper Emulation: Context and Current Status", Digital Preservation Testbed Project Koninklijke Bibliotheek, National Library of the Netherlands, 2003, https://web.archive.org/web/20050305150902/http://www.digitaleduurzaamheid.nl/bibliotheek/docs/white_paper_emulatie_EN.pdf, accessed 17/03/2019.

[4] E. Cochrane, Rendering Matters, Archives New Zealand, 2012, http://archives.govt.nz/rendering-matters-report-results-research-digital-object-rendering, accessed 17/03/2018

[5] E. Cochrane, Visual Rendering Matters, Archives New Zealand, 2012, http://archives.govt.nz/resources/information-management-research/rendering-matters-report-results-research-digital-object-0, accessed 17/03/2018

[6] P. McGlone, "A guy in Minnesota is the museum world's answer to old technology", The Washington Post, 20/04/2018, https://www.washingtonpost.com/entertainment/museums/a-guy-in-minnesota-is-the-museum-worlds-answer-to-old-technology/2018/04/19/78cae5aa-3dcd-11e8-8d53-eba0ed2371cc_story.html accessed 17/03/2019

[7] J. Rothenberg, "Ensuring the Longevity of Digital Information, Council on Library and Information Resources, 1999, http://www.clir.org/wp-content/uploads/sites/6/ensuring.pdf, accessed 3/17/19

[8] E. Cochrane, The Emergence of Digital Patinas, The Digital Preservation Coalition Blog, 2017, https://dpconline.org/blog/idpd/the-emergence-of-digital-patinas, accessed 17/3/2019

[9] K. Rechert, D. von Suchodoletz, R. Welte, "Emulation based services in digital preservation", Proceedings of the 10th annual joint conference on Digital libraries, Pages 365-368, 2010

[10] R. Lorie, "Long term preservation of digital information", Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries, Pages 346-352, 2001.

[11] Dioscuri team, "Dioscuri: Ideas and Key Features", 2007, http://dioscuri.sourceforge.net/dioscuri.html, accessed on 17/03/2019

[12] Keep Project Team, "Keeping Emulation Environments Portable", 2012, https://web.archive.org/web/20120121170030/ http://www.keep-project.eu/ezpub2/index.php, accessed 3/17/19

[13] E.Oltmans, R. van Diessen, H. van Wijngaarden, "Preservation functionality in a digital archive", JCDL '04 Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries, Pages 279-286, 2004.

[14] D. von Suchdoletz, J. van der Hoeven, "Emulation: From Digital Artefact to Remotely Rendered Environments", The International Journal of Digital Curation, Issue 3, Vol 4, 2009.

[15] S. Bonin, "Preservation and Long-term Access via NETworked Services: Keeping digital information alive for the future", 2009, https://planets-project.eu/docs/comms/PLANETS_BROCHURE.pdf accessed 17/03/2019

[16] K. Rechert, I. Valizada, D. von Suchodoletz, J. Latocha, bwFLA - A Functional Approach to Digital Preservation. Praxis der Informationsverarbeitung und Kommunikation 35(4): 259-267, 2012

[17] Scaling Software Preservation and Emulation as a Service Infrastructure, "About EaaSI", 2019 https://www.softwarepreservationnetwork.org/eaasi/, accessed 17/03/2019

[18] T. Liebetraut, K. Rechert. "Management and Orchestration of Distributed Data Sources to Simplify Access to Emulation-as-a-Service". iPRES 2014

[19] OpenSLX GmbH, "EaaSI Group", Gitlab.com, 2019 https://gitlab.com/eaasi, accessed 17/03/2019

[20] Association of Research Libraries et al, "Code of Best Practices in Fair Use for Software Preservation", 2019 https://www.arl.org/focus-areas/copyright-ip/fair-use/code-of-best-practices-in-fair-use-for-software-preservation, accessed on 17/03/2019

[21] Software Preservation Network, "Home" , 2019 https://www.softwarepreservationnetwork.org/, accessed on 19/03/2019

[22] K. Rechert, T. Liebetraut, O. Stobbe, N. Lubetzki, T. Steinke, "The RESTful EMiL: Integrating emulation into library reading rooms", Alexandria: The Journal of National and International Library and Information Issues, Vol 27, Issue 2, pp 120-136, 2017.

[23] F. Moretti (2013). Distant Reading. Verso, London, 2013.

[24] M. Addis, J. O'Sullivan, J. Simpson, P. Stokes, J. Tilbury, "Digital preservation interoperability through preservation actions registries: iPres 2018 – Boston" iPres 2018, Boston, 2018

[25] E. Cochrane, J. Tilbury, O. Stobbe, "Adding Emulation Functionality to Existing Digital Preservation Infrastructure", iPres Conference 2017, 2017

16th International Conference on Digital Preservation
iPRES 2019, Amsterdam, The Netherlands.