Useable Software Forever

The Emulation as a Service Infrastructure (EaaSI) Program of Work

|  |  |  |
| --- | --- | --- |
| **Euan Cochrane** | **Klaus Rechert** | **Jurek Oberhauser** |
| *Yale University Library*  *United States*  [*euan.cochrane@yale.edu*](mailto:euan.cochrane@yale.edu)  *0000-0001-9772-9743* | *University of Applied Sciences Kehl*  *Germany*  *rechert@hs-kehl.de*  *0000-0002-2454-4374* | *OpenSLX*  *Germany*  *jurek@openslx.com*  *ORCID* |
| **Seth Anderson** | **Claire Fox** | **Ethan Gates** |
| *Yale University Library*  *United States*  *seth.r.anderson@yale.edu* | *Yale University Library*  *United States*  *claire.fox@yale.edu* | *Yale University Library*  *United States*  *ethan.gates@yale.edu*  *0000-0002-9473-1394* |

**Abstract – The Emulation as a Service Infrastructure (EaaSI) program of work is dedicated to ensuring all software is usable forever. In this paper we outline why we believe this vision is important, describe the technologies included in the EaaSI software, describe some of the challenges we face in realizing this vision, and finally outline some of the future developments we are working on to expand the impact of the EaaSI program of work.**

**Keywords – Emulation, software preservation, migration, file formats**

**Conference Topics – Community; Exchange; Innovation.**

1. Introduction

“In short, software is eating the world.”

- Marc Andreessen, Wall Street Journal, August 20, 2011 [1]

Software is everywhere: software runs our societies’ businesses, industries, transportation systems, power grids, and healthcare systems. Software is essential to the provision of authentic digital evidence in our legal systems. Scientific research methods and their associated research outputs are increasingly software-dependent, and our economic and cultural heritage, including the records of our long-life economic and defense assets such as buildings, ships, and aircraft, is increasingly born-digital. Notable examples include computer Aided Design (CAD) files, websites, documents, slide sets, emails, time-based media, spreadsheets, databases, design files, project management files, etc. Moreover, born digital objects frequently require specific older software applications in order to be accessed at all, or accessed without meaningful distortion. Without maintaining access to legacy (no-longer supported) software applications, aging software-dependent industrial control and healthcare systems will not be able to be troubleshooted or redeveloped, long-life economic and defense assets won’t be able to be maintained, digital evidence won’t be able to be authenticated, software-dependent research reproduced, or digital cultural heritage accessed.

With the importance of access to software in mind, the Emulation as a Service Infrastructure (EaaSI) program of work is endeavoring to ensure software is always accessible with a dedication to the vision “Usable software, forever”. In this paper we outline why emulation and software preservation are vital to all digital preservation endeavors and what EaaSI is doing to make emulation practical, scalable, and usable for all, indefinitely.

# Why Software Preservation and Emulation?

All digital files are functionally software. Born digital objects are fundamentally different in their nature compared to previous information capture technologies such as paper, film, or velum. While we generally think of born digital objects as the files that software applications create, and which we share between systems, they are functionally much more complex, and these complexities have significant implications for our information management and long-term preservation approaches.

Born digital objects are distributed amongst multiple files, most of which are files provided by the software applications used in the creation of the objects and provision of the objects as ‘information experiences’ or ‘performances’. This characterization of born digital objects as information experiences or performances was first proposed in the National Archives of Australia’s green paper published in 2002 [2] in which they described a “performance model” for digital objects:

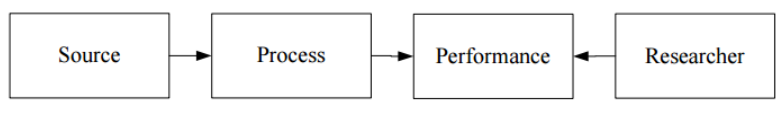


Figure 1: The Performance model

The implications of this model are significant, for example: if a user “opens” a Microsoft word file using a modern version Microsoft Word for Windows, this initiates a complex process in which hundreds of files on the computer provide instructions in response to the results of the processing of the information from the previous files in the process. A change to any one of those files has the potential to change the information presented to the users at the end of the process, change how the user can interact with the “information experience” they are presented with (for example it might make it impossible for the user to check the contents of metadata fields using the application’s User Interface (UI)), or completely terminate the process making the file “unopenable”.

The figures below provide an example in which the same spreadsheet “performance” is created using the combination of a primary data file (the traditional ‘record’) - a .xls file – and the files that represent two different applications. The result of the computer processing the instructions in the two different sets of files (both containing the primary data .xls file) is two different performances with different information being available for interaction in each. In one performance [figure 2], after executing the combination of instructions in the software files and data file a comment is made available in the User Interface (UI) providing important context. In the other performance [figure 3] the combination of instructions in the set of files leads to no comment being visible or available for interaction.

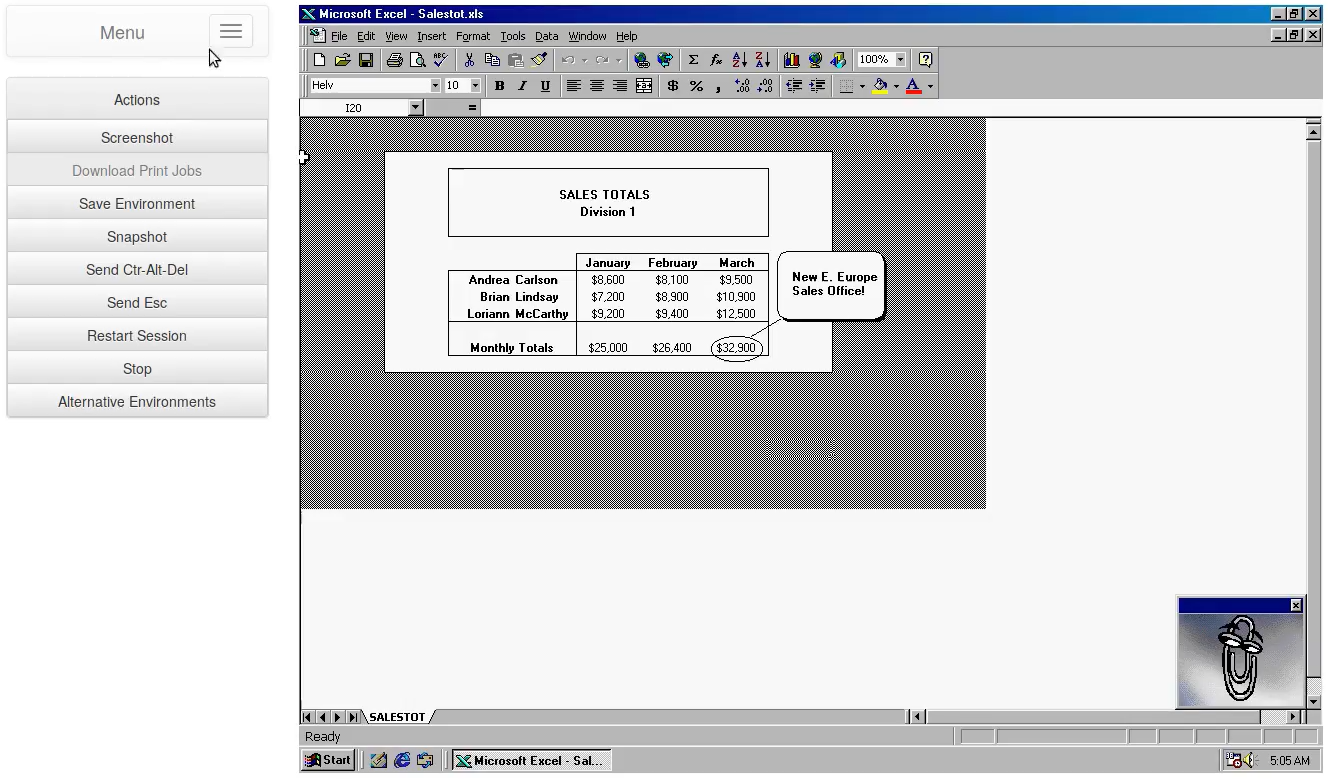


Figure 2: .xls workbook opened in Microsoft Excel for Windows 95

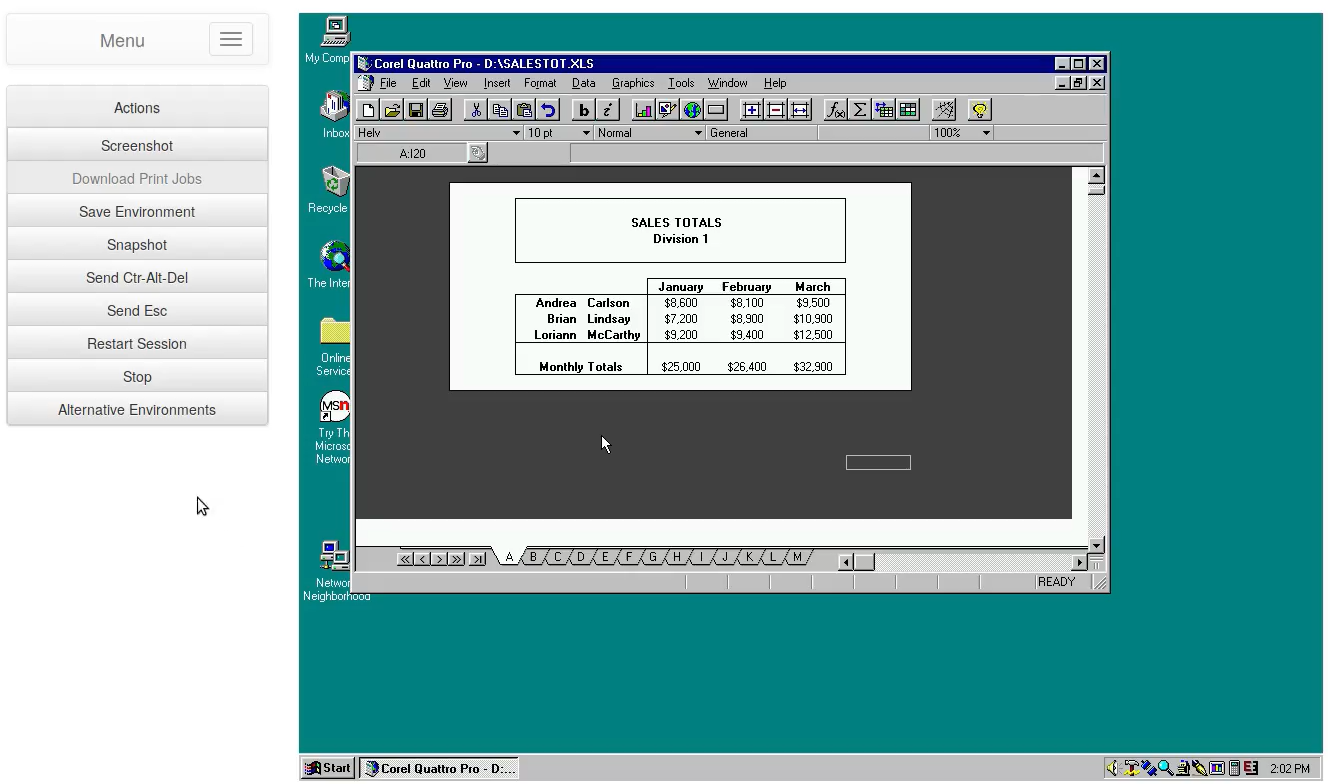


Figure 3: .xls workbook opened in Corel Quattro Pro for Windows 95

In some situations, these outcomes can be even more pernicious. Information that is not visible or even non-existent in the original performance can be added when the software files that made up the original performance are substituted for different files. For example, in the example below we see tags being added to the title of the document that indicate something about the document should be kept private. In the original these don’t exist, but when the primary data file is combined with a different set of software files, the resultant performance includes instructions that interpret parts of the primary data file as requiring the “private” tags to be presented as they are. This is deceptive and would likely not be useful as evidence of the security classification of the document.



Figure 4: A WordPerfect file opened in WordPerfect 5.2 for Windows 3.11

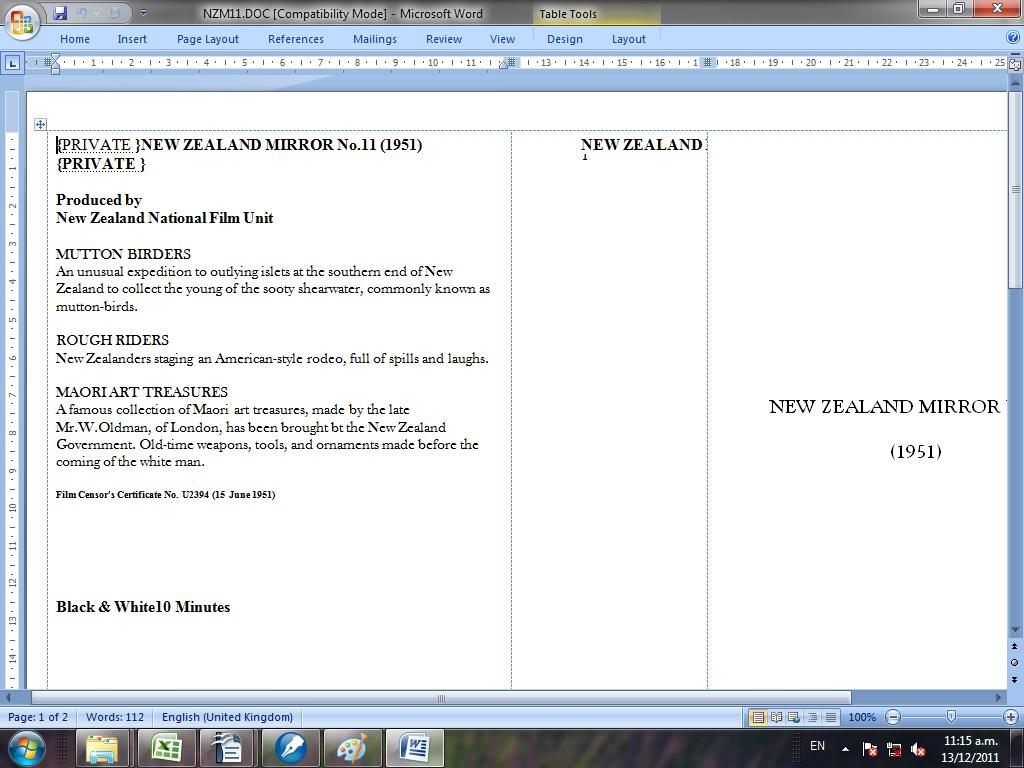


Figure 5: A WordPerfect file opened in LibreOffice 3.3.0

In both of these examples we see that digital objects are made up of many files in addition to the primary data files . All of the files involved provide sets of instructions that are executed by the computer to produce the performance or information experience which users can then interact with. It is trivially true that changing the primary data files is highly likely to result in a change to the resultant performance. This is because the computer will interpret the different instructions in the file(s) differently, producing a different performance. However, as we have seen in the examples above, changing the software files will also. All of the files involved in an object, whether those that are part of the software applications or the primary data file(s) that make up the object, have the same function as part of the object, the function as instructions interpreted by the computer. This being the definition of software [3], it illustrates the general case that, functionally, all digital objects are software. So, to preserve any digital materials in a way that is meaningful and verifiable for users, we must be able to preserve all components of a software performance - the primary data files (e.g. the .doc, .xls, .wpd files), and the files that make up the software applications involved (e.g. the executable files, and many of the other files in the installation directory of each application).

To this end, software preservation and emulation proves valuable in a number of distinct use cases, including the following:

## Providing Verifiable Digital Evidence Requires Software to be Preserved and Accessible

Formal interactions in the modern world are increasingly undertaken using digital technology. From property purchases, to banking, to communicating, to conducting and evaluating research, to legal proceedings, our formal interactions are increasingly ‘born-digital’ (originating in digital form [4]). With the COVID-19 pandemic this trend has only accelerated as we’ve collectively sought ways to continue operating society without having to interact in person. Given the digital nature of these formal interactions, the records of these interactions, and the records of government and society in general, are also now mostly born-digital.

Records are preserved primarily to provide evidence of activities that occurred [5]. To provide trustworthy evidence, records must be verifiable. As we saw in the earlier examples, if we only preserve the primary data files from digital objects, and not the original software application’s files, then we will not be able to provide access to the performances. If we cannot provide access to the performances, it follows that we will not be able to provide access to the digital objects themselves (i.e. what results from the performances, the experience we can interact with). Without the software the objects can change in ways that can become not just incorrect, but potentially deceptive. In addition, the primary data files themselves (e.g. the .xls, .docx, .pdf, .psd files) should be considered functionally to be software. Since the primary data files depend on other software files, and are software themselves, in order to preserve access to trustworthy, verifiable digital evidence, we must preserve access to software.

## Explaining (Selling) the Importance of Digital Preservation and Archiving by emphasizing the age of digital objects

When trying to explain why digital preservation is important it can be quite difficult to connect digital files with value, age, and importance. However, by presenting files in conjunction with legacy software, users much more quickly see and experience the age of the objects, which in turn makes them seem valuable as a consequence of their perceived age. The idea of needing to preserve things that seem “old” is already well-embedded in our cultural consciousness, so visibly emphasizing the passage of time makes it that much more “obvious” to users and our wider stakeholder community that digital preservation is important.

As the EaaSI community has grown, we’ve experienced many instances when showing EaaSI in action where users experience a strong reaction to seeing and hearing “old” legacy software running again. Since the microcomputer revolution of the 1970s, every generation has memories of their first experiences with computers. Sounds like a cassette tape loading a game into a Commodore 64, the Windows 95 start-up sound, the iconic “you’ve got mail” notification, the ubiquitous iPhone default ringtone, and the sound of a Skype call connecting, invoke visceral reactions in many of us. The look and feel of software from the 1980s and 1990s have themselves become so iconic as to provoke the creation a new genre of retro-software interfaces by modern-day fans (e.g. this recent game <https://www.kickstarter.com/projects/yachtclubgames/mina-the-hollower>). This emotive reaction coming from users of EaaSI often helps the EaaSI team and stakeholders when we are subsequently explaining why everything we do in digital preservation and digital archiving is important. Since things from the past are old, there is an assumption that they must need work to keep them accessible. This association between the emotive reaction and the assumption of work required to keep old things accessible, makes the task of explaining (or selling) the value of digital preservation much easier.

## Emulation and Software Preservation are (together) the Only Option and/or the Only Economical Option

There are limited tools available for maintaining access to digital objects as technology changes. The primary candidates for maintaining access to born digital objects are:

### Migration

Migration is a valuable method for ensuring content in digital objects can be reused in modern software. Migration is the process of replicating some content from a digital object performance using a different primary data file (or files) and a different set of software from the original. Normally the new software works on modern computers whereas the original software is considered functionally obsolete when the objects are migrated.

The challenges migration presents to digital archives are at least two-fold. Firstly, most archives retain the original primary data files from a digital object when they migrate content from the object. This means that normally their storage requirements roughly double after the migration has completed. This extra storage requirement has an economic and environmental impact which can be considerable, especially over time. Secondly, it is currently extremely costly to validate the results of a migration process. In research undertaken at Archives New Zealand [16], Cochrane found that it is very difficult to automatically test a migration process due to the difficulty in automatically identifying changes made to an object. This is partly due to the finding that most objects seemed to include at least one rarely used software feature, and so methods that use shortcuts that exclude rare features are not effective at scale. Another reason for this problem was identified as behavioral: some users used software in ways that meant the digital objects could only present their information using that software but did not use features that could be automatically looked for and automatically validated post-migration. This meant that there was no way to automatically/programmatically identify the features in the file to be migrated.

Cochrane also established that manually testing a migration-equivalent process took on average 9 minutes per object. Figure 6 shows a table from Cochrane’s “Rendering Matters” report [17] that extrapolates the time it would take to test the outcomes from migrating various percentages of various numbers of objects:

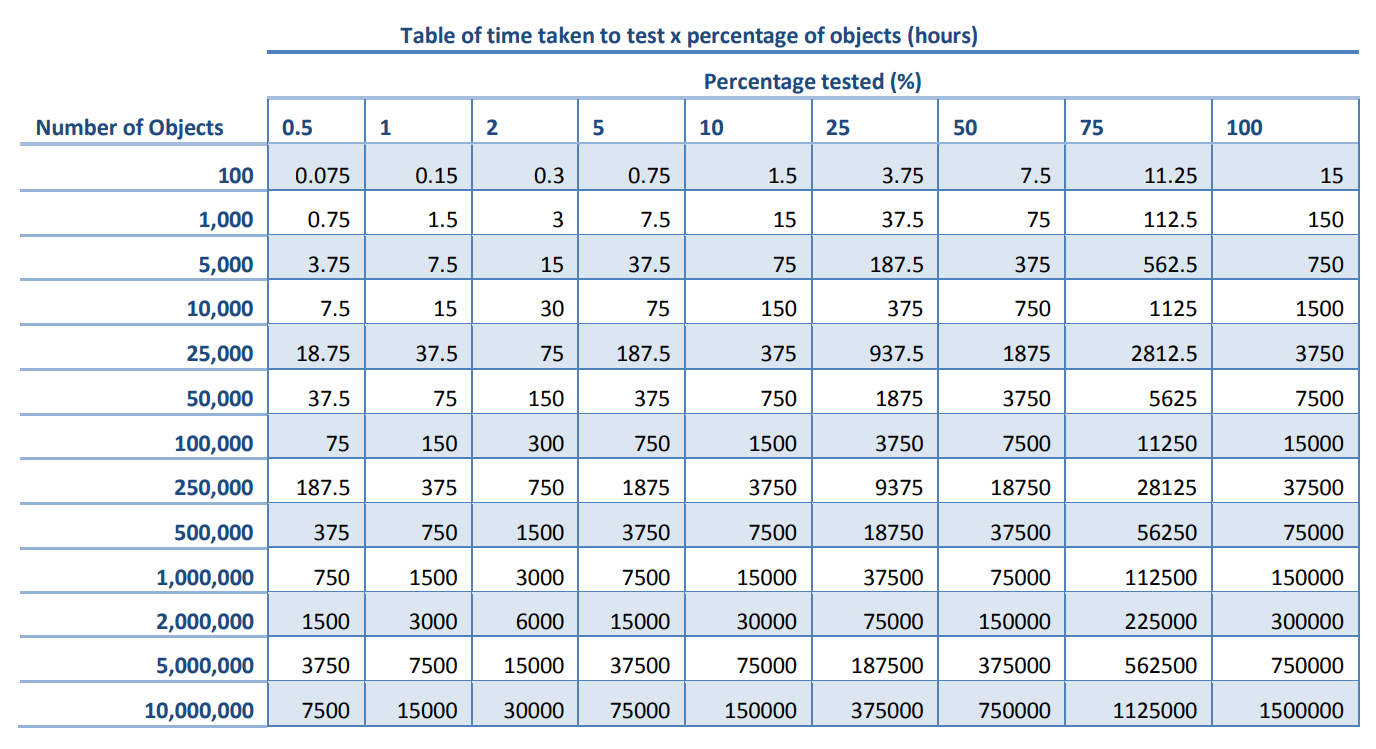


Figure 6: Table of an extrapolation of the time taken to test the migration of digital objects

Many archives now have over a million objects in them. Assuming this table is accurate, to test only 0.5% of 1,000,000 objects would take one person nearly 19 weeks working 40 hours a week with no breaks. Scaling manual testing of migration processes in this way would seem to be very expensive for most archives, and likely uneconomic.

Additionally, many modern “documents” have no format to migrate to. Google Docs documents don’t have a “file format,” for example. Content from them can be exported into files with formats like .docx and .odt but natively they are made up of database entries and the other files that make up the server-based Google Docs software. The only way to preserve these performances (outside of rewriting the software) is to preserve the software that serves the “documents” in our web browsers. In the case of Google Docs, this would likely require preserving the software on multiple servers, including web servers, database servers, file servers and application servers, as these are required to provide the performance that we interact with when we “open” a document in Google Docs. We would also need to preserve access to a compatible web browser.

Finally, cycles of both creation and retention of born-digital objects are unpredictable and archives may have to retain access to migration tools for a very long time if they intend to continue receiving old digital objects throughout that cycle. For instance, the donor of a digital collection may have continued using a particular piece of software long past its "official" support date or “end-of-life" (then donated even longer past). Given that, archives would likely have to use emulation to run migration tools as the tools themselves become functionally obsolete on current computers; it is not clear why an archive that was continually collecting would spend time migrating documents when they would always be able to migrate them just in time for when they are needed.

### Normalization

Normalization is primarily described as undertaking migration on objects as soon as they are received in an archive. This is intended to ensure they are always available in a format that is compatible with modern software. This has all the same issues as migration, if not more so as archives immediately incur the costs of both migration and the preservation of at least two copies if they keep the originals.

1. *Re-Writing of Software*

Re-writing or recreating software so it is compatible with current computers is another option that has been proposed for ensuring access to digital objects over time. However, if we must re-write software to account for every change in file formats over time (as we assume *eventually* every format will become obsolete), this could rapidly become economically unfeasible. Given that there are currently at least 12575 file formats (as documented in Wikidata.org), that this number will only go up over time, and that testing to ensure each format + new-software combination would have to be extremely thorough to ensure the new software doesn’t change the information experience/performance, re-writing software is likely only economically practical when it is applied to re-writing emulators. Re-writing one emulator could ensure access to many emulated computers, which could ensure access to many legacy software applications which could ensure access to virtually unlimited digital objects.

1. *Emulation*

Emulation is the most economically feasible method for ensuring long term access to digital objects because in principle (discussed below) and in practice [18], it scales extremely effectively. As discussed, one emulator can ensure access to many emulated computers which can each ensure access to many legacy software applications which in turn can each ensure access to virtually unlimited digital objects. It’s also possible to preserve emulated computers at one organization permanently, and then share the emulated computers on demand-only when needed. In doing so, the work to create, document, and preserve emulated computers and the legacy software “environments” they provide access to can be distributed such that even small organizations can potentially afford to implement emulation and use preserved software to provide access to their content.

In addition, the work to verify the effectiveness of an emulator can be shared. Computer hardware compatibility verification is a well-established process that has always resulted in differences between specific hardware configurations (e.g., there are minor differences between all individual computers), changes that are accepted. The entire Personal Computer (PC) and software industry is predicated on the assumption that software can be installed on any PC that meets basic “compatibility” requirements. So fortunately, it is only those relatively lightweight requirements that ever need to be met to verify that an emulator is sufficient. This is in contrast with the work to verify the effectiveness of a migration process, which must be tested for every individual object as users don’t expect their objects to change, especially when they are the thing that is meant to be being preserved.

The scalability of emulation and software preservation for digital preservation makes investments in emulation-based solutions worthwhile in most situations. In addition, emulation is a “just in time” method, i.e., it is only used just in time for when it is needed. This means that most of the time, once emulated computers have been created, they can be stored at a few organizations, then accessed by many more, only when they are needed. Furthermore, if future users want to reuse data from preserved objects, the data could still be migrated out into new files by using the original software to do so (if necessary, by chaining multiple applications together to move data between multiple formats). This would have the dual benefit of also enabling users to see what was lost during the migration (by enabling them to compare with the original in emulation) and enabling migration to occur whenever, and only when it was needed. In this scenario the work to verify the migrated data can be undertaken just in time also, further saving resources.

As well as being the most economic option, for a variety of objects software preservation and emulation are the only option for ensuring continued access to digital objects. Microsoft Chart files, for instance, no longer open in any modern application. In addition, games, disk images, and other types of complex digital objects often simply can’t be migrated to new technologies.

## Emulation For Appraising Digital Content

## Until recently, objects that have been received by memory institutions as digital files (either ’born-digital’ or ‘received-digital’ (digitized elsewhere) have often been transferred to the institution on external media. This is beginning to change as organizations find better ways to support network/internet-based transfers [6], but for most memory organizations this is the primary way digital objects are transferred to their organizations and will be for quite some time. In addition, organizations will continue to receive digital storage drives or disk image files made from drives that come from the desktop computers of notable people and likewise for drives from servers from notable services/systems.

## Disk images can be difficult to appraise, especially those representing the drives of entire computers where the context in which the data on them was used was as part of live systems. However, appraisal archivists/practitioners are usually unable to access that context and are limited to (at best) browsing the file systems on the drives to evaluate the value of any specific files or groups of files that they can find. This usually involves either:

## Opening a copy of the disk image (to ensure no changes are inadvertently made to the original) in a disk image review application (like FTK Imager), browsing the file system within the application, and exporting interesting files

## Mounting the disk image (likely in read-only mode) on their local file system and browsing it directly using whatever file explorer application is on their operating system

## They then open the files with whatever "compatible" software is available on the modern computer they are using. Usually this means opening them in the software that the operating system has associated with a particular file extension, MIME type, or type code.

## As we have established, opening files in non-original software can cause changes to be introduced to the information presented by the computer when opening the files. This can mean that when appraising files in the way described above, appraisal archivists may not see the value of something that did have real value. There are also cases where multiple files need to be interacted with together, along with software, in order to view the compound-object that they represent. Many office documents are tied to each other through Object Linking and Embedding (OLE) functionality [7] for example, and do not function or are missing content if all the required files are not present and accessible in the same software application. Emulation allows appraisal archivists to instead open the disk image itself as a virtual computer if it contains an operating system or open the disk image attached to an emulated computer with software installed on it that is contemporary to the content on the disk image. In both cases, the ability for the archivist to understand the content and view it in a meaningful way, is greatly increased.

## Computers and Software Applications are Historic Artifacts

Our focus should not be on digital objects alone, as current research has extended to address the historical notoriety of computer systems and software applications, representing a distinct field of study for which emulation is an essential component.

Computers are the artists’, engineers’, programmers’, authors’, and regular workers’ toolkits of the current age. There exists a huge opportunity to archive, preserve, and make accessible these toolkits for future generations to not just view and interact with, but to reuse to create new outputs in the future in much the same way some artists use very old techniques to create their art. Without the ability to preserve software, or the ability to emulate old computers, preserving computer environments as artifacts will be impossible.

Many software applications are historic artifacts. From Microsoft Office’s ‘Clippy’, to voting machine software, from minesweeper to Minecraft, software applications have had and continue to have a huge impact on society, and for this reason alone should be preserved for posterity.

# Why EaaS

For the average user, obtaining a legacy software application can be very difficult, and once obtained, legacy software can be challenging to install, authenticate, configure, and operate. Older applications are frequently unable to function on modern operating systems, and even when the requisite operating legacy systems can be found, they in turn, are unlikely to function, or function well, on contemporary computer hardware. The problem is only increasing as our computing hardware continues to advance.

Emulators solve many problems, by allowing users to easily run legacy software on modern computers. Emulators are themselves software applications, applications that simulate one computer on another computer, allowing users to install and use software on the simulated computer. An emulated computer is a computer that is simulated or “emulated” using an emulator software application. Emulators are most often used to simulate computers that have a “hardware architecture” that is different from the computers that the emulators are being run on. This allows the user of the emulator to run software that is compatible with the emulated hardware but not compatible with the hardware that the emulator is running on.

Despite solving many problems, emulators also come with many challenges. Emulation technologies can be difficult to employ and particularly challenging to employ at scale. Emulators often require specialist expertise to use and are non-standard tools that information technology departments rarely support. These barriers to the large-scale use of emulators have been addressed by the “bwFLA Emulation as a Service (EaaS)” framework [8].

The EaaS framework enables emulated computers to be made more easily accessible and allows, via only a web browser, for seamless access to the software running within the emulated computer. With EaaS, users do not need to understand how to configure an emulator to ensure it runs a particular operating system. Instead, the EaaS framework provides templates for pre-configuring emulators to support a wide range of operating systems.

In addition, EaaS provides a way to save storage space when scaling the use of emulators for different applications that have similar dependencies. With EaaS users can create “derivative” computing environments (or “environments” – the term we use to refer to both the emulated computer and the software installed on its virtual drive) that are created by saving changes made to an existing environment in a separate file from that which stores the main environment. When the new environment is then re-run it uses both the derivative-environment file and the source environment file at run-time to provide the full environment experience. In doing so this saves the user from having to save copies of all the data that would be the same between two environments that are only different in a small way (e.g. one may have an additional application installed on it).

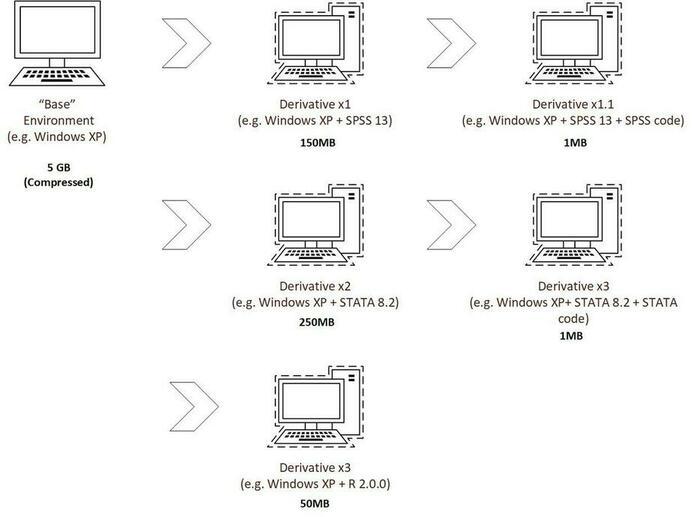


Figure 7: In EaaS, derivative environments can build on dependencies from base environments without the need to save redundant data.

# Why EaaSI

In starting the EaaSI program of work we recognized that the EaaS framework had the potential to enable the wide-spread use of hardware emulators and pre-configured “software environments”, at scale in cost-effective and efficient ways, and thus to become an invaluable resource for digital preservation efforts worldwide. However, EaaS does not alone solve the problem of finding software, or of the work required to configure and document all the software applications that we need to preserve. To do this the digital preservation community needed to make legacy software easy to find, and to share the work to configure and document emulated computers and the software applications they run. EaaSI was developed to fill this gap and to make the use of emulators and legacy software for access, easy.

EaaSI’s primary goal is to enable the scaling of access to both emulation technology and the use of legacy software for providing access to digital objects. To address the latter, we have established the open source EaaSI software and an EaaSI network in North America (with an additional nascent EaaSI network recently starting in Australia [9]). The North American network currently contains sixteen members, primarily at large research Universities. The EaaSI software is built on EaaS and makes use of the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) to enable users to request, share and synchronize metadata between nodes (installations of the EaaSI software) participating in an EaaSI network. This allows one organization to find a software application, install it, configure it, and document it, then publish its metadata to the network. All other nodes in the network can regularly harvest published metadata so local staff-users can see what is available across the network. Staff-users then choose to replicate emulated computers (or “environments”) from other nodes and run them locally to accomplish a goal. For example, a staff-user may replicate an environment containing a Computer Aided Design (CAD) software application in order to provide access to a CAD object that a patron requested. The staff member does not need to configure the emulator (thanks to EaaS), or find, configure, or document the software (thanks to EaaSI). Instead, they may simply import their digital object, load it into an existing emulated computer, configure the file to open in compatible software, and save their work as a new derivative environment. This saves a significant amount of time and resources and makes using both emulation and legacy software to accomplish practical tasks, simple for the staff member.

EaaSI also provides a new interface for EaaS in which users can more easily discover, access and document software and computer environments. By making it easier to document software, we hope to ensure that it is easier for future users to use in the future when interface paradigms have changed.

Using these approaches of sharing and simplification, we are aiming to make the process of accessing content using original legacy software so seamless that users forget that it is complicated and expect legacy software to be available and usable whenever they need it. Users have the option to fully configure all aspects of the software and emulation tools themselves and thorough documentation is included with EaaSI [10]. However, our goal is that such configuration is always optional and normally unnecessary. To fulfill this vision we’re working to turn EaaSI into common infrastructure, shared and used by organizations everywhere.

# Building services on the Emulation as a Service Infrastructure

We’re working to enable legacy software to be seamlessly integrated into any long-term access and re-use contexts, to become core-information management infrastructure. To do this we are working to build out EaaSI’s Application Programming Interfaces (APIs), to create tools using those APIs, and to facilitate a community-based development process. Focus on these strategies is intended to both solve problems and add to EaaSI’s practical use right now, but also inspire others to build on the EaaSI infrastructure to solve problems in additional domains.

## The Universal Virtual Interactor (UVI)

The UVI is a tool we have developed on top of the EaaSI infrastructure to provide the ability for users to click on a link to a digital object (for example in a library’s catalogue or an archival finding aid) and have it automatically open in a representative version of the “original'' software, within their web browser, using an emulator. The name (and the use of "virtual" despite it also/primarily using emulation in addition to virtualization) is an homage to the Universal Virtual Computer (UVC) concept developed by IBM and the National Library of the Netherlands (Koninklijke Bibliotheek, KB). The UVI is intended to be "universal" and (theoretically) work with any files/digital objects. It’s called an interactor not a "viewer" or "renderer", as it's not just for "rendering" or "viewing". Rendering and viewing are primarily passive activities, but digital object experiences are not passive, they’re interactive. We want to be able to enable users to interact with their digital objects presented as an experience that is as close to the “original” as possible. That interaction might include such things as turning on and off “track changes” functionality in a document, viewing embedded metadata through standard application menus, browsing and submitting queries through database interfaces, interrogating and temporarily changing spreadsheet formulae or embedded scripts, etc.

With further development the UVI could be integrated with discovery and access platforms and configured to give users the choice of which software to use to complete their digital object performances. Further they could select a file from a catalog or finding aid, have it matched to multiple environments in EaaSI by the UVI, then choose which environments to use to complete the information experiences/performances. This can all be completed on-demand, just in time for when the user requests it.

## EaaSI Virtual Reading Rooms

EaaSI is designed to allow staff to add files to an emulated environment and securely share access to the environment with one or more users. EaaSI can provide a dedicated access page for the environment, or an environment can be embedded on any arbitrary/custom page via HTML. This process could be used with legacy software environments that have many applications on them to provide access to multiple digital objects at once using a single environment. We are working to refine the process, particularly for providing access to secure materials, as a “Virtual Reading Room” service so that it could be seamlessly integrated with existing discovery and access platforms, services and workflows. In this increasingly remote-working context we hope this toolset will prove particularly attractive, not just for providing access to older objects using legacy software, but potentially for providing secure access to any restricted digital content.

*EaaSI Community*

While design and desired functionality for various pieces of EaaSI tooling may at this point be well-articulated (see above), converting emulation and EaaSI into core services requires constant, open, critical feedback from its intended user community on implementation. The team regularly convenes and solicits input from current and prospective users, investing in and facilitating paths of communication (recurring calls with representatives of the North American EaaSI network; an online Community Forum and issue tracker with registration open to all[11]) to ensure both that EaaSI services address real-world needs and workflows, but also that the program remains tapped in to challenges potentially beyond the scope of EaaSI tooling alone (see below).

# Challenges

## Legal/Copyright

Copyright law provides the biggest challenge to scaling EaaSI globally. In the United States many of us are fortunate enough to be able to rely on the rights defined in the Copyright Act that are described in the Code of Best Practices in Fair Use for Software Preservation [12]. The code describes the legal grounds upon which the EaaSI network participants are operating. However the situation in the United States is unfortunately not replicated globally, and even what exists in the US, is not ideal. There continue to be challenges in the United States in a number of areas of copyright law including the need to circumvent DRM in order to maintain access to usable software (despite recent progress with Digital Millennium Copyright Act exemptions). Making progress in the area of copyright law will be one of the continuing challenges all EaaSI users will need to focus on in coming years.

## Lack of Emulators

EaaSI is relying on existing open-source emulators developed primarily by volunteers. We’re very grateful for their work and are able to provide access to many different types of emulated computers as a result. However, there are still gaps in our library of emulatable computers. Over time these gaps may grow unless additional emulators are created and integrated into the EaaSI framework.

## Distributed Digital Objects

As discussed, more and more objects are becoming increasingly distributed across networks and the internet. Modern CAD/BIM designs are often dependent on files spread across multiple servers and computers with embedded references that are easily broken. Online web services often rely on databases and other files stored on multiple different computers (“servers”) that together present a dynamic information experience to users on request. Modern video games are often sold solely as digital downloads, are constantly updated with new patches (which makes it hard to track their versions over time), and often have to be constantly connected to remote servers in order to be played. Mobile applications are often little more than a front end to various web-services and will become non-functional when those services go away. Some work has been done to address this by prototyping methods for preserving web services [13], and our work to add the ability to network devices in emulated networks will help to provide some infrastructure that could help address these issues.

## Integration

To make EaaSI successful at scale it needs to be seamlessly compatible with many different types of systems. At the most immediate level, EaaSI needs to be able to integrate with digital preservation/digital repository systems so it can retrieve disk images and content from them for management and access. It also needs to integrate with access and discovery platforms to enable environments provided by EaaSI to be made available directly to users via these existing platforms.

Integration with access and discovery platforms presents new challenges however. As we work to provide access to entire databases, web servers, and desktop environments from notable individuals using EaaSI, we will need to develop methods for indexing and documenting the content in them so they can be made discoverable through existing discovery tools and systems. Once discoverable there will be demand to enable direct linking into content that exists inside of environments made accessible by EaaSI, something that will be relatively challenging due to a lack of generic ways for achieving this across different computing platforms and legacy operating systems.

Finally, we will need to be able to easily get data in and out of legacy systems running in EaaSI, and to connect modern computers to services running in EaaSI via the legacy APIs. For example, users will likely want to be able to connect remotely to database servers running in EaaSI in order to be able to query them from modern applications. Fortunately, the querying protocols for many of these scenarios still have modern implementations that could enable that, but making secure connections to legacy computers running in EaaSI will require new features, and will need to be made as user-friendly as possible due to the potential for the knowledge for how to make this function fading over time.

## Application Signatures and a Registry for Software

The National Archives of the United Kingdom (TNA) has done the world a great service with its creation and support of both the PRONOM file format registry and the DROID file format identification tool. These are widely used and acknowledged as some of the most important tools within the digital preservation community [14]. Being able to identify file formats is particularly important for undertaking migration as a strategy as it helps with matching migration tools to files. There is an equivalent approach that could be beneficial for an emulation strategy. By automatically identifying the interaction applications of objects using “application signatures” we could match primary data files to their interaction applications. Currently though, application signatures to feed to a tool like DROID or Siegfried do not exist, and even more critically, there is no central registry for software applications that could provide us with identifiers to match to. PRONOM does include a very limited set of software metadata and Wikidata.org includes a great deal more. However, neither are likely appropriate as homes for signatures for interaction applications, nor for the information we might want to share publicly about software environments that include the applications. In other words, there is a gap here. A gap we will aim to fill in the future.

In the meantime, we are working to include information about software and environments that we’re adding to EaaSI into the Wikidata.org database. This data will then be able to be made available to digital preservation systems via the Preservation Action Registries API [15].

# The Future

EaaSI is not static. There is still ongoing conceptual and technical development of EaaSI, and likely will be forever. With a mission of “Useable software, forever” EaaSI will need to evolve as the software landscape evolves. In the near term this will include needing to move past primarily supporting the emulation of single personal computers to other domains such as networks of computers, mobile devices, server machines, and whatever comes next. In this section we outline some of these initiatives and describe how we’re moving the EaaSI platform forwards to meet the needs of the changing digital landscape.

## Mobile

Mobile computing has taken over the computing world over the last couple of decades. There are more mobile devices than personal computers and their cultural, scientific, and economic impact continues to be hugely historically important. For these reasons the EaaSI team has recognized the need to be able to preserve and provide access to mobile operating systems, applications, and the files and data they support. We are in the process of adding an emulator for Android-based devices to the EaaSI platform as a first step to exploring how best to address the challenges of preserving the mobile computing universe. Preserving mobile computing experiences provides many challenges, from replicating the experience of “app stores”, to re-creating or simulating the various network services that mobile applications rely on, to simulating output for the myriad of sensors that mobile devices have built into them, and finally to providing meaningful replicas of the experiences of interacting with the huge variety of physical devices that have made up our mobile universe since the early 2000s. By starting small we hope to provide a testing ground for trying different options for addressing this growing list of challenges.

## Networks

Computers have been connected in networks since at least the late 1950s. Home computer users began networking their machines to others at scale with the wide availability of dial-up modems in the 1980s which led to services like Bulletin Board Systems (globally) and Minitel (in France), and finally to the internet that we all use today. Networks of computers and devices have become so prevalent and ubiquitous that we no longer think twice about the complexity involved in just connecting our phones to our local wireless networks when returning home. However, networks are complex. They involve multiple computers and services running on them that have to be orchestrated together to function appropriately. With EaaSI we are working to enable users to create emulated “networks” made up of multiple emulated computers that are connected to each other and can share information between them over standard networking protocols. These may include complex research environments, email servers, database management systems, and more. We have a functional prototype of this software that we are currently in the process of adding to the core EaaSI platform. This will be another transformative tool for our users. It will open a new set of potential use cases for the application of EaaSI to ensure long term access to large scale systems, enterprise databases, functional interactive websites and more.

## Automation

The task to configure and document legacy software is increasingly urgent as knowledge of how legacy software works and how to configure it is fading from institutional memories as those who used it leave the workforce. Increasing complexity and technological change means that in the future we’ll have to have ways to enable users to interact with software that has interfaces that they’ve potentially never experienced. Even where software has been configured and basic documentation created, it is often difficult for users to accomplish tasks, especially in complex enterprise systems such as those used in government and industry. To address this, we are developing methods to record interactions in EaaSI and play them back with specified parameters. This will allow future researchers to accomplish tasks using software running in EaaSI (such as finding all information about a user in a database) just by clicking a labeled button or calling an API function.

We have also developed a functioning prototype of a tool to automatically record metadata about compatible file formats using two different methods: 1. Optical Character Recognition (OCR) applied to the interfaces of graphical environments in EaaSI and 2. using the Windows Operating System programming interface to read the menu items and pass them out of the emulator. This can be used to automatically record metadata about the format’s applications can open, save, export, and import, further reducing the manual effort needed when adding new applications to an EaaSI network.

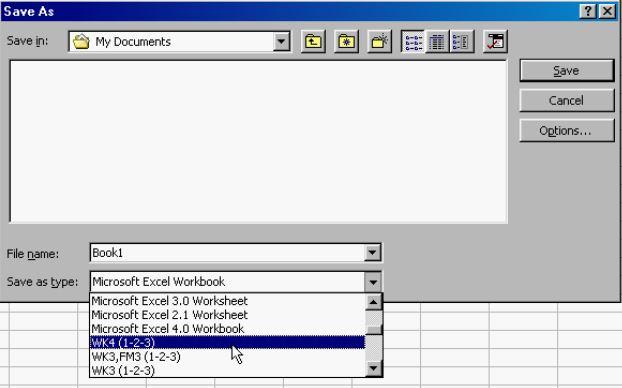


Figure 8: An example of the software application format compatibility metadata that we have developed a tool to automatically document.

We are further working to make use of this metadata to revive the idea prototyped in the PLANETS project of enabling automated migration of content between digital objects using emulation (“migration by emulation”) [16]. This will allow on-demand migration of content from legacy digital objects into new objects that make use of modern software. It will make use of the original software to migrate content to a more open or newer format and this process could potentially be run multiple times in a “chain” process to create objects that are fully functional on modern computers.

# Conclusion

Ensuring software is usable forever will be an ongoing challenge. However, it is essential for ensuring society’s knowledge, culture, and history persist over time. While we’re focused on using software to ensure content is unchanged over time, we also believe that regular reuse of content in archives helps keep both the materials and the organizations that steward them relevant. Enabling the use of emulation at scale will ensure that we can continue to migrate some content out of old digital objects and into new ones, at scale. This in turn will ensure the extractable content stays usable and therefore relevant to whomever can benefit from it in the future.

With the Emulation as a Service Infrastructure program of work we are building and freely sharing infrastructure; infrastructure that we hope is not just useful, but inspiring. We imagine a world in which digital content is always usable, regardless of the software that is part of it. To get there we are going to need others to build on what we have started. With the first EaaSI services: the UVI and the virtual reading room functionality, we hope that we are showing some of the potential for how EaaSI could be used at scale to solve long term preservation and information management issues. We don’t know what else might be achievable with this infrastructure, but we’re excited to find out.

REFERENCES

1. *Why Software Is Eating The World. M. Andreessen*. Wall Street Journal. August 20, 2011. Retrieved 8th of March 2022 <https://www.wsj.com/articles/SB10001424053111903480904576512250915629460>
2. *An Approach to the Preservation of Digital Records.* H. Heslop, S. Davis, A. Wilson. 2002. National Archives of Australia. https://www.naa.gov.au/sites/default/files/2020-01/An-Approach-to-the-Preservation-of-Digital-Records.pdf
3. *Software*. Wikipedia Contributors. 8th March 2022. <https://en.wikipedia.org/w/index.php?title=Software&oldid=1071678404> (last visited Mar. 8, 2022)
4. *Born-digital*. Wikipedia Contributors. 8th March 2022. <https://en.wikipedia.org/w/index.php?title=Born-digital&oldid=1064529162> (last visited Mar. 8, 2022)
5. *Record*. The Society of American Archivists. <https://dictionary.archivists.org/entry/record.html#:~:text=Dearstyne%201992%2C%201Records%20are,provide%20lasting%20evidence%20of%20events>. (Last visited Mar. 8, 2022)
6. *Exactly: A New Tool For Digital File Acquisitions.* B.Lyons. AVP. January 13, 2016. <https://blog.weareavp.com/exactly-a-new-tool-for-digital-file-acquisitions> (last visited Mar. 8 2022)
7. *Object Linking and Embedding.* Wikipedia Contributors. https://en.wikipedia.org/w/index.php?title=Object\_Linking\_and\_Embedding&oldid=1049945261 (last visited Mar. 8, 2022).
8. *bwFLA - A Functional Approach to Digital Preservation*. PIK - Praxis der Informationsverarbeitung und Kommunikation 35(4):259-267. November 2012. DOI:10.1515/pik-2012-0044
9. *Australia will be getting its own Emulation as a Service Infrastructure (EaaSI) network soon* [Online forum comment]. M. Swalwell. 2nd February 2022. <https://groups.google.com/g/australasia-preserves/c/gaKLhFQTALE/m/7CkB-FH6BAAJ?utm_medium=email&utm_source=footer>
10. *The EaaSI Handbook*. The EaaSI team. 2022. <https://eaasi.gitlab.io/eaasi_user_handbook/> (last visited Mar. 8, 2022)
11. *EaaSI Communtiity Forum.* <https://forum.eaasi.cloud> (last visited June 23, 2022)
12. *Code of Best Practices in Fair Use for Software Preservation*. Software Preservation Network. Revised 2019, <https://www.softwarepreservationnetwork.org/wp-content/uploads/2020/02/2019.2.28-software-preservation-code-revised.pdf> captured at <https://perma.cc/FK5K-MWJR>.
13. *Preservation strategies for an internet-based artwork yesterday, today and tomorrow*. C. Roeck, R. Gieschke, K. Rechert, J. Noordegraaf. Procedings of the 16th International Conference on Digital Preservation iPres 2019, p179 . <https://ipres2019.org/static/proceedings/iPRES2019.pdf> DOI 10.17605/OSF.IO/GF2U9
14. *“And it's official! PRONOM wins the #DPWorldCup!”.* Tweet message. The Digital Preservation Coalition Team. Twitter. 25 February 2022. <https://web.archive.org/web/20220225145245/https://twitter.com/dpc_chat/status/1497222900482457602>
15. *PAR Overview*. PAR team. Accessed 8th March 2022. <https://parcore.org/>
16. *Demo: Migration-by-Emulation*. I Valizada, D. von Suchdoletz, K. Rechert. 2011. Proceedings of the 8th International Conference on Preservation of Digital Objects: iPRES 2011 - Singapore. P248. <https://phaidra.univie.ac.at/o:294293>
17. *Rendering Matters.* E. Cochrane. Archives New Zealand. January 31, 2012. <https://web.archive.org/web/20130218111126/http://archives.govt.nz/rendering-matters-report-results-research-digital-object-rendering>
18. *(Re-)publication of Preserved, Interactive Content – Theresa Duncan CD-ROMs: Visionary Videogames for Girls*. D. Espenschied, et al. Proceedings of the 12th International Conference on Digital Curation. Chapel Hill, NC: University of North Carolina, School of Information and Library Science, 2015. P233/241. <https://phaidra.univie.ac.at/view/o:429524>