Evaluating Assisted Emulation for Legacy Executables

Swetha Toshniwal Geoffrey Brown Kevin Cornelius
Indiana University School of Informatics and Computing
Gavin Whelan Enrique Areyan

ABSTRACT

Access to many born-digital materials can only be accomplished economically through the use of emulation where contemporaneous software is executed on an emulated machine. For example, many thousands of CD-ROMs have been published containing proprietary software that cannot be reasonably recreated. While emulation is proven technology and is widely used to run both current and obsolete versions of Windows and Unix operating systems, it suffers a fatal flaw as a preservation strategy by requiring future users to be facile with today's operating systems and software.

We have previously advocated "assisted emulation" as a strategy to alleviate this shortcoming. With assisted emulation, a preserved object is stored along with scripts designed to control a legacy software environment and access to the object enabled through a "helper" application. In this paper we significantly extend this work by examining, for a large data set, both the cost of creating such scripts and the common problems that these scripts must resolve.

1. INTRODUCTION

This paper describes a project to develop practical techniques for ensuring long-term access to CD-ROM materials. The underlying technology for our work consists of off-the-shelf emulators (virtualization software) supported by custom automation software. We use automation to capture the technical knowledge necessary to install and perform common actions with legacy software in emulation environments and hence mitigate a fundamental flaw with emulation. This work directly addresses issues of sharing through networked access to emulators and object-specific configuration and assistance.

Over the past 20 years CD-ROMs were a major distribution mechanism for scientific, economic, social, and environmental data as well as for educational materials. Our work has primarily focused upon the nearly 5,000 titles distributed by the United States Government Printing Office (GPO) under the Federal Depository Loan Program and thousands more distributed by international agencies such as UNESCO. Recently, we have expanded our study to the thousands of commercial titles held by the Indiana University Libraries. In the short-term these materials suffer from physical degradation which will ultimately make them unreadable and, in the long-term, from technological obsolescence which will make their contents unusable. Many such titles (as much as 25% of the GPO titles and perhaps more for commercial titles) require execution of proprietary binaries that depend upon obsolete operating systems and hardware. A widely discussed technical strategy is to utilize emulation (virtualization) software to replace obsolete hardware. [6, 2, 4, 11, 10, 7, 3, 5] Recent surveys of issues related to the use of emulation in preservation based upon lessons from the Planets project include [9, 12].

A fundamental flaw with this approach is that future users are unlikely to be familiar with legacy software environments and will find such software increasingly difficult to use. Furthermore, the user communities of many such materials are sparse and distributed thus any necessary technical knowledge is unlikely to be available to library users. The work described in this paper is aimed at alleviating these issues.

As mentioned in the abstract, we have previously proposed a strategy of "assisted emulation" which attempts, through the use of helper applications and scripting, to simplify access to legacy materials. [13] In prior work we described a simple pilot study aimed at determining the basic viability of this approach. In this paper we significantly expand this work with an emphasis upon understanding the issues and difficulty associated with creating the scripts required by our strategy. In particular, we describe the results of a study involving several hundred CD-ROMs, both government and commercial through which we are able to make recommendations about the basic emulation environment, additional software requirements, and a collection of techniques used to automate access to the individual titles.

2. REVIEW OF ASSISTED EMULATION

Our approach, as described previously in [13], relies upon storing scripts along with legacy materials which are executed automatically by a "helper program" when a user accesses the materials through an ordinary web browser. For a given digital object, a single "click" causes the associated script(s) to be downloaded and executed to start and configure an emulation environment on the users workstation. This approach is illustrated in Figure 1. Where a user requests access to an object through a browser on the client machine to a web server (1). The web server responds with a signed applet (2) causing a helper program on the client machine to execute a local emulator (3). This emulator is configured using scripts stored on the network to execute software and access objects also stored on the network. This model makes certain basic assumptions which we elaborate upon in the sequel. First, the emulator and its basic operating environment are accessible from the client machine; and second, the preserved object and any necessary scripts are accessible on networked storage.

Throughout our work we have assumed that, for a given

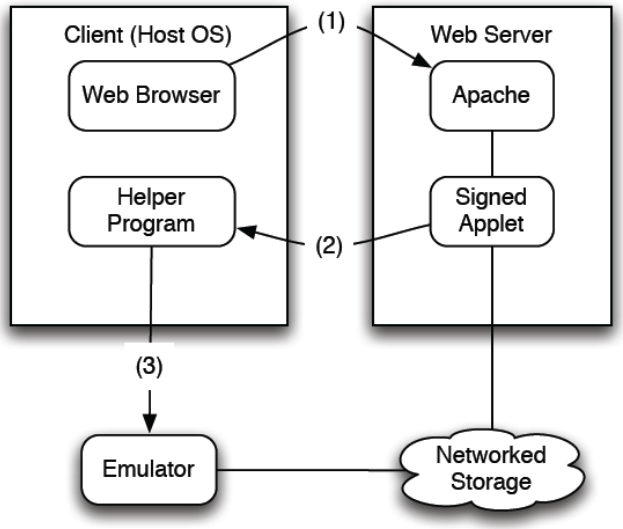

Figure 1: Client Request for Networked Resource

platform (e.g. PC, or "classic Macintosh"), most software can be accommodated by a small number of configurations. For example, when working with legacy digital objects originally supported by various versions of Windows, we have found that Windows XP provides a high degree of backwards compatibility and hence a large fraction of digital objects from prior releases of Windows can be accessed by an emulator running Windows XP. Thus, we assume that the client machine has access to an emulator with a small number of reference configurations. In this paper we concentrate upon preserving objects designed to execute under Windows and MS-DOS – in separate work we have investigated the use of emulation to preserve classic Macintosh applications. [1]

Our "reference" configuration consists of Windows XP and Adobe Reader, with other software (e.g. Office and QuickTime) installed as needed by the helper scripts. It is not feasible to combine all additional software in a single reference image because some CD-ROMs depend upon specific software versions; however, the results we present suggest that a limited set of configurations could be created to minimize the frequency with which run-time installation of helper applications is required.

As mentioned above, we assume that the digital objects and necessary scripts are accessible through networked storage (in our work we have used the Andrew File System (AFS)). [8] The objects we work with are the thousands of CD-ROMs in the Indiana University Libraries. Our archive organization is illustrated in Figure 2 which shows how uniquely numbered CD-ROM images (ISO files) are stored in eponymous directories along with item specific scripts (e.g. install.exe), and generated ISO files containing required helper software. The CD-ROM images are created with standard software from the physical disks and the scripts are created using a process described in Section 4.2. This figure differs from our previous work with the inclusion of additional ISO files to provide helper applications required by specific CD-ROM images.

3. RESEARCH ENVIRONMENT

Over the past five years we have built a research collection of nearly 5000 CD-ROM images from materials in the Indiana University Libraries. These include United States Government documents, publications of international organizations (e.g UNESCO) and foreign government, commercial publications, and educational publications. In our initial work on assisted emulation we focused upon the US Government documents which generally have minimal requirements for additional software and offered limited variety in terms of installation processes. We have greatly expanded this work and in this paper provide results based upon analyzing 1325 CD-ROMs of all types.

For our emulation platform we use VMWare Server running on top of Windows 7 (64-bit) (we also run on top of Linux in our development environment). There are many alternative emulation platforms for executing Windows operating systems with appropriate APIs enabling automation. Within the virtual machine, we run Windows XP Professional with a basic set of application programs.

Assisted emulation depends upon the creation of scripts that can be executed when a patron requests access to a particular CD-ROM image. For Windows emulation we use the freeware tool AutoIt [1], a BASIC-like scripting language that facilitates automating user tasks by capturing keystrokes, mouse movements, and window controls. While we installed AutoIt on our baseline machine, it is only required for the creation of scripts which can be conveniently converted into executable files. This is discussed further in the next section.

4. CD-ROM AUTOMATION

Our automation work consists of two phases for each CD-ROM image. In the first phase we explore the image by mounting it in our reference virtual machine to gather some

[1]AutoIt Automation and Scripting Language. `http://www.autoitscript.com/site/autoit/`

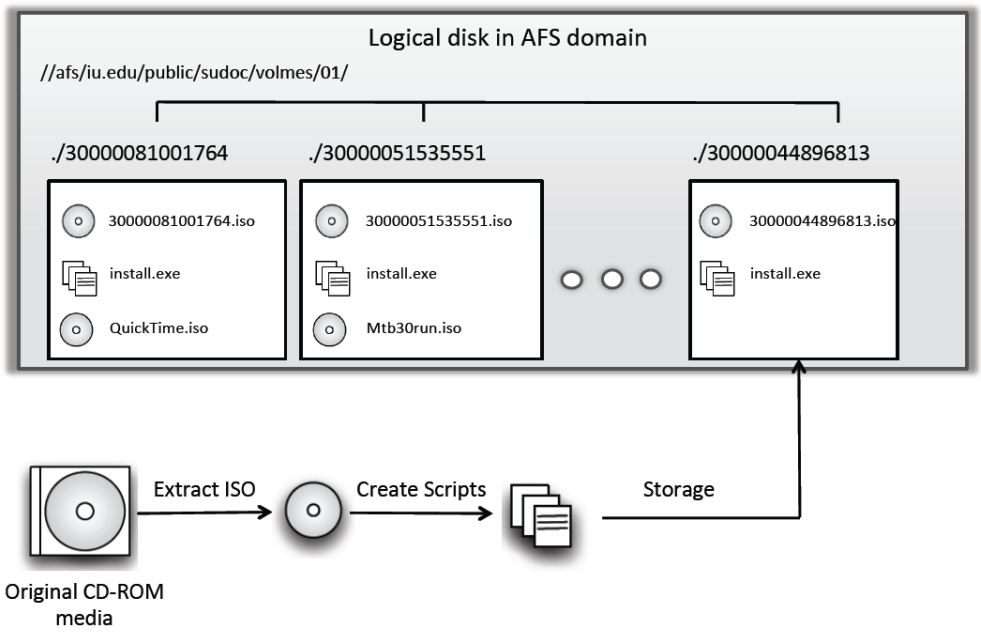

Figure 2: Organization of the Virtual Archive

basic information including: whether the image includes executable software, any required installation procedures, dependencies upon other CD-ROM images (in some cases a set of CD-ROMs are interdependent), and whether there appear to be additional software requirements. Once the requirements of a CD-ROM image are understood, the second phase consists of automating any installation processes and creating ISO files containing external software that may be required – as our repertoire has grown, we have found there is a considerable amount of reuse both in terms of scripting and these additional ISO files.

4.1 Exploration Phase

The exploration process can be quite simple – for example, many CD-ROMs contain README files that explain installation procedures and define externally required software. However, in some cases there is little or no guidance provided. Furthermore, many of the CD-ROMs we considered were in foreign languages which our research team could not readily read. Where external software is required, it is frequently difficult to determine which versions are compatible both with the CD- ROM software and our execution environment (e.g. some CD-ROMs require 16-bit QuickTime and only the last 16-bit version works correctly on Windows XP). Once the necessary software version is determined, it can be a challenge to find a copy (e.g. Multimedia Toolbox 3.0).

One of the more vexing problems we ran into was dealing with foreign languages – especially Asian. There are two aspects to this problem – our inability to read the language and the need for foreign language support in Windows. Resolving this problem typically required: (1) determining the appropriate language, (2) for east Asian languages installing Windows support, (3) configuring the appropriate language option in Windows.

We found it most efficient to install language support as part of our base image meaning that only steps (1) and (3) are necessary on a per-CD-ROM basis. In order to execute some programs provided on images it was necessary to configure various compatibility modes in Windows XP. These include changes to virtual memory settings, changing to 16-bit color, and setting file properties for specific compatibility modes. For programs designed to run in DOS mode, some images also required configuration of extended memory (XMS).

An additional complication was dealing with objects which were published on multiple CD-ROMs where there are cross-disk dependencies. For example, a program on one image might require access to a file on another image. Our current strategy is to simultaneously mount all dependent disks. This has a known limitation – VMware can only support up to three simultaneous virtual CD-ROMs. Ultimately, we may need to develop a more sophisticated helper program which will help the user to selectively mount CD-ROM images from a given set.

In summary, exploring the CD-ROM images revealed the program requirements, special cases, and required the development of strategies to handle these special cases. However, these problems are not unique to assisted emulation – even if patrons were provided access to the original CD-ROMs and machines capable of executing them, somebody would have to understand these obsolete environments sufficiently to overcome any obstacles. With assisted emulation there is at least the possibility to capture the required knowledge in scripts.

4.2 Helper Scripts

As mentioned previously, we use AutoIt for our script development. AutoIt executes simple programs in a BASIC-like language. Furthermore, the program provides a tool to

```
Run("D:\SETUP.EXE")
WinWait("Setup")
ControlClick("Setup", "", "Button1")
WinWait("", "successfully installed")
ControlClick("", "successfully installed", "Button1", "", 2)
WinWait("CD-ROM Delos")
ControlListView("CD-ROM Delos", "", "SysListView321", "Select",
ControlListView("CD-ROM Delos", "", "SysListView321",
"FindItem", "Delos"))
ControlSend("CD-ROM Delos", "", "SysListView321", "!{ENTER}")
WinWait("Delos Properties", "Shortcut")
WinClose("CD-ROM Delos")
ControlCommand("Delos Properties", "Shortcut", "SysTabControl321","TabRight")
WinWait("Delos Properties", "Compatibility")
SendKeepActive("Delos Properties", "Compatibility")
Send("{TAB}{SPACE}")
ControlClick("Delos Properties", "Compatibility", "Button11")
ControlClick("Delos Properties", "Compatibility", "Button10")
Run("D:\WIN\DELOS.EXE")
```

Figure 3: Example Script

convert these programs into small executables (.exe files). In general, most of the complexity of these scripts comes from handling special cases – for example, setting compatibility mode for older Windows programs. Consider the script in Figure 3. The basic setup is accomplished within the first 6 lines. The remainder of the script is concerned with setting compatibility mode for the installed executable, and then running that executable.

Through the exploration of 1325 disks, we have built a script catalog to deal with the commonly occurring special cases. Some more difficult cases include autorun CD-ROMs where the `autorun.inf` fails under Windows XP, handling international CD-ROMs where window names and commands are in non-English Unicode, and installations where restarting the virtual machine is required after the software is installed. With experience, we have developed techniques to deal with these and other challenging cases. In general, we have been able to reuse script fragments to deal with many of the issues that arise in practice.

4.3 Analysis

In selecting test cases for this work, we have attempted to choose materials from a wide range of genres, languages and publication dates (1990-2010) and have analyzed 1325 CD-ROM titles. To understand the breadth of these choices, consider the Table 1 which provides a sampling of the range of these characteristics.[2] Any such characterization is, by necessity, somewhat arbitrary. We distinguish between commercial publications and government publications because our experience with the government printing office materials suggests that many government publications are primarily data in a limited set of formats; although, some earlier publications required installation of proprietary programs. Our selection of "genre" categories is intended to illustrate the breadth of materials – i.e. these are not all data sets. The language category is the least ambiguous. Note the relatively high fraction of Asia languages; this isn't too surprising given the source of the materials – a major research library. However, it also illustrates a challenging problem for archivists of such materials as installation of these disks presents a potential language barrier.

These various categories of works have widely varying software requirements. As mentioned previously identifying and finding additional software has been a major issue for this project – this is discussed further in the sequel. The work described in this paper has had its share of failures – CD-ROMs which we have not yet succeeded in executing. Finally, a key objective of this project has been to evaluate the cost of employing the strategy we are advocating. We present preliminary results on these costs.

4.4 Additional Software

In some cases, additional software requirements could be determined by the file types present on a CD-ROM, in other cases, error messages received when attempting to execute a CD-ROM provided helpful hints. It is not always necessary to find the exact version of software requested by CD-ROM documentation. For example, we found Adobe Reader to have excellent backwards compatibility – so good that we have added Adobe Reader X to our base configuration. In the few cases where images required an earlier version, our scripts uninstall Adobe Reader X and then install the required version. In other cases, the installation process requires specific versions (e.g. Office 97 or Office 2000) where one would expect better backwards compatibility. QuickTime has extremely poor backwards compatibility and it is essential that the correct version be selected. Finally, we sometimes substitute alternative packages; for example, we used Adobe Reader in place of Abapi reader (a Chinese "clone" of Adobe Reader). The Table 2 summarizes the software we installed based upon the CD-ROM requirements. The percentage of CD-ROMS requiring each software product is also provided. Unfortunately, determining acceptable version requirements for additional software is a trial and error process. Of the 1325 scripts we consider in this article, 1194 required the installation of some additional software; however, the majority can be satisfied by an enhanced baseline image including Adobe Reader, Internet Explorer, and Microsoft Office.

[2]This table is based upon an initial analysis of 240 CD-ROMs.

Category		Genre		Language	
Commercial	84.0%	Periodical/Serial	33.0%	English	49.2%
Government	16.0%	Informational	14.5%	Japanese	21.3%
		Historical	6.8%	Chinese (PRC)	14.6%
		Database	5.6%	Chinese (Taiwan)	7.0%
		Educational	5.6%	German	5.5%
		Media	5.2%	French	4.3%
		Cultural	4.4%	Hungarian	4.3%
		Bibliography	4.0%	Czech	3.5%
		Entertainment	3.6%	Romanian	3.5%
		Geographic	3.6%	Bulgarian	2.7%
		Geological	2.8%	Estonian	2.7%
		Academic	2.4%	Greek	2.7%
		Survey	1.6%	Italian	2.7%
		Literature	1.6%	Polish	2.7%
		Biographical	1.2%	Slovanian	2.7%
		Agricultural	0.4%	Korean	0.8%
		Political	0.4%	Russian	0.8%
		Statistical	0.4%	Spanish	0.3%

Table 1: Characteristics of CD-ROMs

Program	Number	Percent
Adobe Reader	695	58%
Internet Explorer	255	21%
QuickTime	108	9%
Microsoft Office	65	5%
Windows Media Player	49	4%
Java	13	1%
Photoshop Pro	6	$< 1\%$
Real Audio	2	$< 1\%$
Multimedia Toolbox	1	$< 1\%$

Table 2: Additional Software

Based upon these results, we recommend a virtual machine configuration supporting three CD-ROM drives (the limit for IDE), Adobe Reader X, and the Windows International Language Package. In most cases, Microsoft Office should be installed, but as mentioned above, there are situations where the latest version is incompatible. Thus, it may make sense to support a CD-ROM collection with a small number of base images (e.g. both with and without Office) in order to simplify run-time customization.

4.5 Failures

We have not yet succeeded in executing all of the CD-ROM images. A small number had physical errors introduced during the imaging process – this can be addressed by creating fresh images. More challenging are several CD-ROMs created for early versions of Windows (e.g. 3.1) which we have not been able to execute on either Windows XP or Windows 98. We have not yet attempted setting up a Windows 3.1 environment to test them. Unfortunately, the scripting tool we use is not available for Windows 3.1 so it is unlikely that we will achieve full automation for these cases. However, the virtual image requirements are likely to be quite small for Windows 3.1 and this may be a case where storing a custom virtual machine image along with the CD-ROM image makes sense.

4.6 Temporal Costs of Scripting

In this section we consider the per-item costs associated with setting up an assisted emulation environment. Factors setting up VMware and creating base images, which are one-time costs, are not considered. Throughout this work we have monitored the time used to create each script. As might be expected, as our team became familiar both with the scripting tool and solved some of the common automation problems, times have declined significantly. The time taken to write a script has ranged from a few minutes to 3 hours with an average of 15 minutes. The data for 1325 scripts are provided in Figure 4. Indeed, virtually any script that took longer than average meant that we encountered some challenge, often for the first time. Examples of the problems that we had to overcome include: changes to environment settings, finding required external software, language issues including support and documentation, installation of multiple programs, installation requiring multiple OS restarts, cryptic error messages, unusually lengthy and complex installations.

Notice that most of these issues are fundamental – they are due to issues with the underlying software and are not due to the scripting process. Some of these issues resulted in specific actions in the scripts (duplicating actions required for a normal installation). Among the more common cases were: language change (12%), Computer restart[3] (13%),

[3]Language changes also require a restart

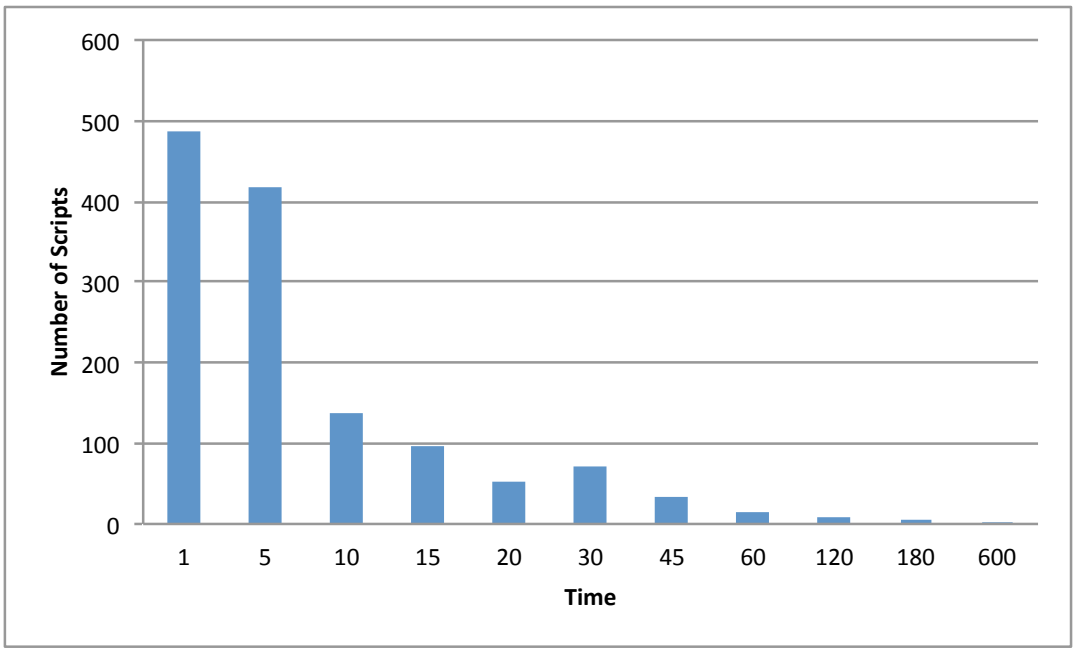

Figure 4: Number of Scripts by Creation Time (minutes)

Java installation (1.5%), free virtual memory (1.1%), display settings change (0.1%), Compatibility Settings (0.4%), and XMS memory configuration (0.1%).

4.7 Script Complexity

Another way to measure the difficulty of scripting is to consider the length of scripts. In Figure 5 we provide a chart of script lengths. The shortest scripts were a single line, for example:

```
Run("D:\start.exe")
```

which hardly justifies a dedicated script ! A more typical script, such as that illustrated in Figure 3 requires some interaction with the CD-ROM installer as well as initialization of an environment for the end-user. This example is 21 lines whereas our average script was 27.5 lines. Many of the longest scripts involved either rebooting the virtual machine during installation, changing the platform language (e.g. to support Asian languages) or installing multiple additional software applications. For example, the 158 scripts that performed language changes averaged 52 code lines. An additional 14 scripts required rebooting and averaged 68 code lines. The longest scripts which did not involve a reboot, also altered system properties (e.g. colors) to create a compatible environment for software designed to execute on older platforms.

As mentioned previously, many of these installation "tricks" are reusable – indeed they are recorded in our scripts. Consider, as an example, a fragment of a script that reboots the virtual machine during installation as illustrated in Figure 6. The key idea is that there are two phases – "prereboot" and "postreboot". The first phase performs the basic installation (mostly elided) and, through the "_RunOnce" procedure marks a suitable variable in the registry. The postreboot procedure starts the installed application.

```
If not FileExists(...) Then
        _prereboot()
Else
        _postreboot()
EndIf

func _prereboot()
        Run(''D:/SETUP.EXE'')
        ...
        _RunOnce()
        Shutdown(2)
EndFunc

Func _RunOnce()
      ...
      If @Compiled Then
          RegWrite(...)
    Else
          RegWrite(...)
    EndIf
EndFunc

func _postreboot()
      ...
EndFunc
```

Figure 6: Script Performing Reboot

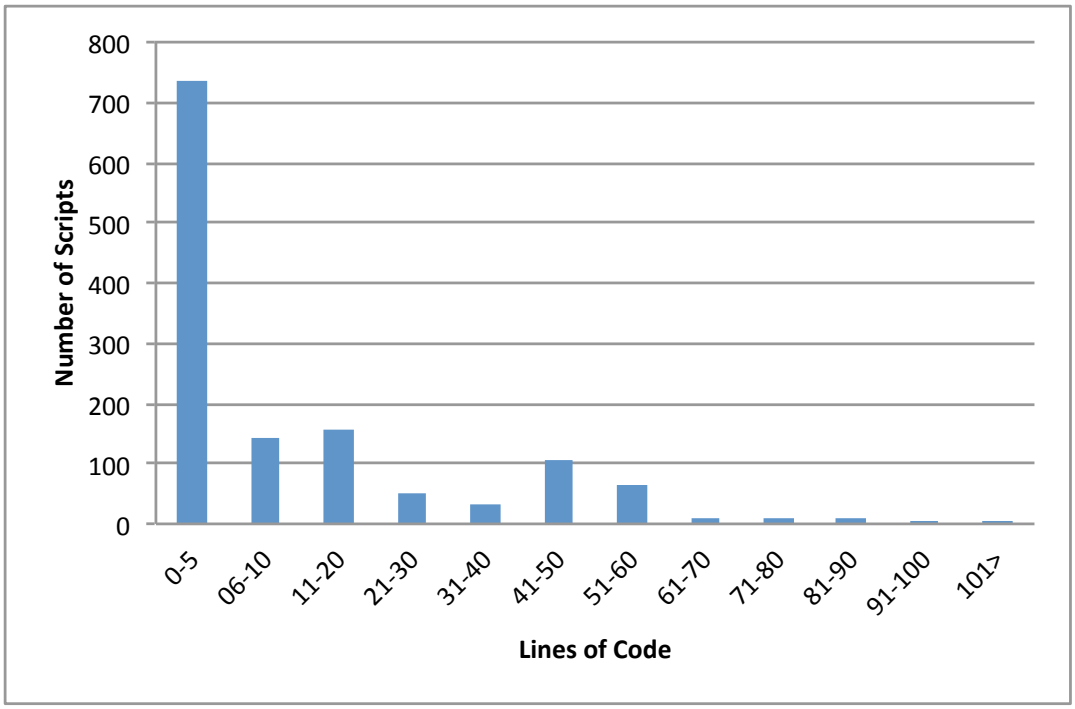

Figure 5: Lines of Code for Scripts

5. DISCUSSION

Clearly the creation of scripts is an extra layer of work required beyond installing and configuring software. The alternative would be to store a separate virtual machine image for each preserved object. For Windows XP these images are 4-8GB which implies a substantial overhead for a 500MB CD-ROM. In contrast with the development of install programs for arbitrary machines with arbitrary software configurations as is required for the development of commercial software, our scripts are required to work only in the tightly controlled environment of a virtual machine image. Furthermore, we have not found the temporal cost of writing scripts is a large additional burden. In a separate project we studied the emulation of CD-ROMs published for "classic Macintosh" machines. In that case, storing customized virtual machine images imposes a much smaller overhead (these images are typically 128MB). [1]

For many in the preservation community, the fundamental questions are how expensive is this approach and what skills are required. Most of the script development was performed by Computer Science undergraduates working as research assistants. These are bright students with some degree of programming sophistication. The data we have presented suggest that, on a per-item basis, an average of 15 minutes is required. In a more realistic production environment with the overhead of developing proper documentation and additional testing, it is reasonable to budget an hour per-item. The actual time requirements of creating the images is quite small (less than 10 minutes per item).

A side benefit of this project is that the process of creating scripts has helped us understand and collate both the common installation problems and the additional software required to preserve CD-ROM materials. In this sense, the creation of install scripts represents only an incremental effort over any principled preservation strategy.

We assumed from previous work that Windows XP would be an adequate platform for emulation of most CD-ROMs created for Windows and MS-DOS operating systems. This has proven to be largely correct; however, as we have noted, we have encountered a handful of CD-ROMs that are so tightly tied to Windows 3.1 that we have not (yet) succeeded in executing them in the Windows XP environment.

The work described in this paper is part of a larger project which aims to create open-source tools to support assisted emulation and which will greatly expand the set of test cases from those we have discussed. We plan to make all of the data, scripts, and helper code available at the end of the project.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant No. IIS-1016967. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

6. REFERENCES

[1] G. Brown. Developing virtual cd-rom collections: The voyager company publications. In *iPRES2011 8th International Conference on Preservation of Digital Objects*, 2011.

[2] S. Gilheany. Preserving digital information forever and a call for emulators. In *Digital Libraries Asia 98: The Digital Era: Implications, Challenges, and Issues*, 1998.

[3] M. Guttenbrunner, C. Becker, and A. Rauber. Keeping the game alive: Evaluating strategies for the

preservation of console video games. *The International Journal of Digital Curation*, 5(1):64–90, 2010.

[4] A. R. Heminger and S. Robertson. The digital rosetta stone: a model for maintaining long-term access to static digital documents. *Communications of AIS*, 3(1es):2, 2000.

[5] Keeping Emulation Environments Portable. `http://www.keep-project.eu/expub2/index.php.\AccessedSeptember2011`.

[6] A. T. McCray and M. E. Gallagher. Principles for digital library development. *Commununications of the ACM*, 44(5):48–54, 2001.

[7] P. Mellor. CaMiLEON: emulation and BBC doomsday. *RLG DigiNews*, 7(2), 2003.

[8] OpenAFS. OpenAFS, 2010. `http://www.openafs.org`. Accessed November 2010.

[9] K. Rechert, D. von Suchodoletz, and R. Welte. Emulation based services in digital preservation. In *Proceedings of the 10th annual joint conference on Digital libraries, JCDL*, pages 365–368, 2010.

[10] J. Rothenberg. An experiment in using emulation to preserve digital publications. Technical report, Koninklijke Bibliotheek, July 2000.

[11] J. Rothenberg. Using emulation to preserve digital documents. Technical report, Koninklijke Bibliotheek, July 2000.

[12] D. von Suchodoletz, K. Rechert, J. Schroder, and J. van der Hoeven. Seven steps for reliable emulation strategies -solved problems and open issues. In *7th International Conference on Preservation of Digital Objects (iPRES2010)*, 2010.

[13] K. Woods and G. Brown. Assisted emulation for legacy executables. *The International Journal of Digital Curation*, 5(1):160–171, 2010.