

PRESERVING LONG-TERM ACCESS TO UNITED STATES
GOVERNMENT DOCUMENTS IN LEGACY DIGITAL FORMATS

Kam A. Woods

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Department of Computer Science

Indiana University

October 2010

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Doctoral
Committee

Geoffrey Brown, Ph.D.
(Principal Advisor)

Beth Plale, Ph.D.

Michael Gasser, Ph.D.

August 20, 2010

Sandra Kübler, Ph.D.

Copyright © 2010

Kam A. Woods

ALL RIGHTS RESERVED

To Mom, Dad, and Majka.

Acknowledgements

My gratitude goes out to all of the individuals who have made my time at Indiana University so wonderful, both intellectually and personally. I would particularly like to thank my advisor, Geoffrey Brown. Dr. Brown has provided indispensable support and guidance to me both as his Research Assistant, and in my continuing growth as a member of the larger academic community. Dr. Brown's special topics class on long-term digital preservation provided the initial motivation for the work presented here. His keen insights into identifying and pursuing novel solutions to problems in this area have provided a foundation for more than three years of rewarding collaboration.

Many thanks are due to the other members of my doctoral committee. Michael Gasser's reputation with former students was instrumental in my decision to come to Indiana University, and his support and guidance during the dissertation process have been invaluable. Sandra Kübler's enthusiasm and insightful comments have been extremely helpful. Both Mike and Sandra have been instrumental in my successful completion of a Computational Linguistics minor as part of these studies.

Valkyrie Savage assisted in the development of a software prototype in the summer of 2007. Her analytical mindset, enthusiastic problem-solving, and comments on design provided valuable

feedback. Stewart Howard assisted on code revisions for the migration system and Mitchell Lutz generated many of the automation scripts used in the virtualization project.

The technical and systems staff of the Indiana University Computer Science department are consummate professionals. Rob Henderson, Bruce Shei, and TJ Jones have quickly and unfailingly solved numerous hardware, software, and storage issues. Lynne Crohn has provided extensive access to legacy Windows software.

Kam Woods

PRESERVING LONG-TERM ACCESS TO UNITED STATES GOVERNMENT DOCUMENTS IN LEGACY DIGITAL
FORMATS

Over the past several decades, millions of digital objects of significant scientific, economic, cultural, and historic value have been published and distributed to libraries and archives on removable media. Providing long-term access to these documents, media files, and software executables is an increasingly complex task because of dependencies on aging or legacy hardware and software. This is a persistent problem for both digital libraries and long-term digital archives, where mandates to maintain and improve access can be overshadowed by ongoing technical and administrative costs associated with digital collections.

There are several widely accepted techniques used by the archival community to preserve materials originally held on legacy media: bitstream preservation, migration of documents from aging formats to modern ones, and emulation for legacy executables. I demonstrate how these techniques can be combined to provide high-quality access to digital collections without compromising long-term archival processes or increasing risk. I show that most technical risk to preserving and accessing legacy born-digital documents can be effectively managed through the careful application of existing open source tools paired with some custom software. I focus on the collection of Government Printing Office documents held on legacy optical and magnetic removable media at the Indiana University Libraries. This collection contains millions of born-digital objects (documents and software) in hundreds of formats.

I present a systematic approach to transferring bit-identical filesystems from legacy media to modern storage, ensuring future operation within legacy environments and supporting integrity checks and deduplication tasks. I describe reliable, high-performance techniques for automated identification, feature extraction, migration, rendering, and distribution of the documents and software contained in this collection. I examine methods that exemplify best practices for providing Web access to digital collections, including high-performance indexing, generation of and access to machine- and human-readable metadata, on-demand migration and rendition of legacy documents, and the construction of a “virtual filesystem” to simplify navigation of the digital archive. Finally, I examine the relationship between these techniques and the development of quantifiable measures of risk for legacy digital objects.

Contents

Acknowledgements	v
Abstract	vii
1 Introduction	1
1.1 Digital Preservation of Removable Media Collections	4
Preserving Government Documents	6
1.2 Approach and Contributions	7
Preservation Issues and the State of the Art	9
1.3 A Preservation Testbed for Removable Media Collections	11
Migration, Emulation, and Rendering	14
1.4 Chapter Outline	17
2 Digital Preservation Overview	20
2.1 Introduction	20

Defining Digital Preservation	21
2.2 Digital Object Models	22
Reference Model for an Open Archival Information System	23
2.3 Bit Preservation and Removable Media	26
2.4 Metadata	27
Metadata Standards	27
Preservation Metadata	29
2.5 Format Identification and Verification	31
2.6 Migration and Emulation	32
Migration	33
Emulation	34
Migration and Emulation in Practice	37
2.7 Managing Risk	38
Existing Work	39
Future Directions	40
3 Media Transfer and Bit-Level Preservation	42
3.1 Origin and Longevity	43
3.2 Imaging Removable Media	45
Verification and Common Errors	47
3.3 Filesystem Processing	50

4	Collection Processing, Metadata, and Access	55
4.1	Testing the Collection	58
4.2	Accessing the Collection	60
	Architecture	60
	Implementations	63
	Metadata Creation	65
	Access Within ISO Images	68
4.3	Access Overview	70
5	Migration	73
5.1	Introduction	74
5.2	System Overview	76
5.3	Document Migration	78
	Format Identification	79
	Migration Tools	82
	Performance Issues	88
6	Emulation	92
6.1	Emulation and Access	93
6.2	Approach	95
6.3	Software	99

Overview	99
Installation Scripts	101
Scripted Data Extraction	103
Linking	104
Emulation Platforms	105
6.4 Discussion	106
7 Conclusion	108
7.1 Discussion	108
7.2 Future Directions	110
Risk Management	110
Preservation Quality Metrics	112
Reducing Rendering and Migration Risks	114
7.3 Closing Comments	115
A Managing Risk	117
A.1 A Case Study in Legacy Document Font Use	118
The Font Substitution Problem	119
Background	122
Fonts in Windows and Word	124
Font Matching Experiments	126

Font Substitutions	131
Discussion	135
A.2 Future Directions	137
B ISO Verification, Automated Digital Object Migration, and Emulation Wrappers	139
B.1 Overview	139
B.2 ISO 9660 Truncation Test Tool	139
B.3 Migration Automation	143
B.4 Emulation Automation	144
Bibliography	150

List of Tables

5.1	Conversion candidates	82
5.2	Conversion success rates	85
5.3	Conversion resource usage	89
A.1	Font Data	127
A.2	Times Variations. Percentages calculated from number of times each font variation is encountered.	130
A.3	Font Pitch	130
A.4	Font Family	131

List of Figures

1.1	Web service for distributed access to legacy documents.	12
1.2	Analyzing rendering performance in an emulated environment.	13
1.3	Automated document migration with OpenOffice.	15
2.1	OAIS Document Object Model	24
2.2	OAIS Functional Model	25
2.3	PREMIS Data Model	30
3.1	ISO 9660 filesystem organization	51
3.2	Common imaging utilities	54
4.1	Testing GPO CD-ROM access.	59
4.2	Example “File System”	62
4.3	Example “File System View”	63
4.4	Namespace	63
4.5	Metadata handling	64

4.6	Example METS Record	66
4.7	System overview	70
4.8	Metadata for a collection item.	71
4.9	Browsing within an ISO image.	72
5.1	User navigation of objects within an ISO image.	77
5.2	Top 18 file distributions by type.	81
5.3	Migration strategies.	84
6.1	Automated launch and configuration of a VM	96
6.2	Organization of the archive	97
6.3	Emulation platform selection	100
6.4	Client request for a networked resource	101
6.5	Reconstructed view of legacy database.	104
7.1	Format risk analysis.	111
A.1	TI83plus font samples	120
A.2	Barcode Rendering	121
A.3	Font Usage for Glossary Documents	128
A.4	Font Usage for Government Documents	129
A.5	Word plug-in identifying substituted “Glasnost Light” text.	135

Introduction

Rapid obsolescence of digital hardware, media, file formats, and associated software interpreters has resulted in an impending crisis: our ability to access and correctly render digital materials of historic, cultural, and scientific interest is increasingly under threat, while the needs of a fundamentally information-driven society demand its continued preservation and availability. Accelerating technological change, limitations of archival infrastructure and associated software technologies, and a growing understanding of the complexities of managing digital collections have all played a role in the development of what is known as *long-term digital preservation*.

In this thesis I develop and explain methods and technologies that can be used to reduce or eliminate specific risks to obsolescent digital holdings while improving facilities for access. This work is based on projects and experiments developed over a period of three years to build a preservation-aware, web-accessible collection from CD-ROMs published by the Government Printing Office and held at the Indiana University Libraries. The methods I employ – bitstream preservation of filesystems from removable media, migration of legacy file formats to modern formats, emulation for legacy executables, and the creation of interoperable metadata – are accepted archival practices consistent with existing mandates for the preservation of government documents [23, 78]. The core

contribution of this work is the demonstration of these technologies being used in a concert with a collection management and access model constructed as a “virtual filesystem”, where distinct but semantically related objects within the collection can be presented together within an access context independent of the underlying storage and processing mechanisms. By employing multiple preservation techniques the design avoids assumptions about how these materials might be used in the future and reduces the likelihood of information loss through a single channel of failure. For current access, the use of both migration where possible and emulation where necessary provides increased flexibility.

Digital preservation research and development focuses on mitigating the effects of changing technologies, and providing error-free (or error-limiting) mechanisms for storage, retrieval, and interpretation of digital collections. Preserving our electronic records and the knowledge they encode is important because these materials are often part of a public trust. In short, we would not have produced and stored them unless we believed they had some inherent value – both to ourselves for ongoing reference and to our ancestors as what Henry Gladney refers to as “the history of the future” [36]. Problematically, this value depends not only on our ability to simply retain the digital object whole and unchanged, but to decode and access the information it contains.

As long-term digital preservation implies long-term *access*, strategies employed to address these problems must take into account not only how to archive “born-digital” materials, but also how to ensure rapid and accurate search, retrieval, and rendering performance over time. “Over time” often reflects an unspecified period; we may identify certain collections as having particular cultural value, but we cannot always anticipate which documents or digital objects will be considered historically significant in the future. Consequently, digital archives must provision for active management of collections in the face of multiple cycles of changing technology, both internally and externally. The need for this is simply stated by Andreas Stanescu in early development on the

INFORM risk assessment methodology:

It is cheaper and safer to analyze and compare potential actions before actions are actually taken, and making a poor preservation decision today can lead to content loss or the need to engage in expensive salvaging efforts later [102].

Archives often employ sophisticated protocols to ensure consistent and reliable performance over all tasks, both technical and administrative, that are applied to these materials. While the protocols themselves may comply with an agreed-upon standard, implementations can diverge significantly. In the seminal report “It’s About Time - Research Challenges in Digital Archiving and Long-term Preservation” sponsored by the National Science Foundation and the Library of Congress, a simple statement elucidates a fundamental issue with bringing practical implementations to bear on archived materials:

Accelerating rates of data collection and content creation and the growing complexity of digital information resources tax current preservation strategies designed to archive simple and self-contained collections of data and documents [43].

This is of particular concern for documents produced by various branches of the United States government, as complex and interrelated datasets and historic materials necessary to support ongoing scientific inquiry, policy developments, and public works represent assets that in many cases increase in value over time – while simultaneously being subject to increased risk of technical obsolescence. Archives thus face significant problems beyond the initial acquisition, assessment, and cataloging of such data. As the useful lifespan of – or mandate for access to – many of these materials may be indefinite, planning of archival storage capacity and scalability are critical. Data provenance, technical and descriptive metadata, document transformation and replication, persistent management, development and description of schemas and hierarchical information for archival organization, and exchange of heterogeneous forms of data between archives all form part of the technical and curatorial processes that are applied to these holdings [43].

1.1 Digital Preservation of Removable Media Collections

Digital preservation efforts contend with numerous risks, including the handling of failure-prone legacy media, risks intrinsic to the digital objects contained on these media, risks inherent to object migration and access environments, risks associated with loss of environmental and rendering context, and risks due to ongoing archival activity. I describe these fundamental issues in further detail below.

Legacy removable media. Physical media used to distribute digital content during the past several decades, including magnetic and optical media, are inherently fragile and degrade with use. However, the problems associated with removable media extend beyond their fragility. Floppy disks, at the time of writing, are essentially historical artifacts; simply finding machines that have the appropriate drives and controllers to read these media can be difficult. Eventually, they will no longer be found except in a museum context. Ultimately, even pieces of well-preserved access hardware will fail. This problem can be exacerbated when institutions depend on “archival-quality” media. An archival-quality CD-ROM may have a projected lifespan of more than a century (under ideal storage conditions), but this is irrelevant if we no longer have the hardware to read and the software to interpret this media within twenty years. Constructing lossless techniques to transfer and preserve the filesystems and digital objects contained on these media to modern storage – techniques that preserve all properties of the media necessary for long-term access – is a process that draws heavily on low-level knowledge of both media and filesystem design. Prior to the creation of any network-accessible path to archival materials, an organization with removable media holdings must first retain or acquire both the technical expertise and the hardware necessary to transfer these materials to modern storage systems. Holdings on removable media must be matched to existing digitally stored metadata (cataloging and bibliographic information) and – potentially – physically instantiated metadata (printed manuals, cases, sleeves, and other accessories which may contain

technical information) for storage in a modern digital archive. Transfer and verification of the bitstreams from these holdings is generally time-consuming and introduces risk of information loss. Checksums and other validation mechanisms may be absent or unreliable, and identification of and comparison to duplicated holdings at other institutions can be time-consuming and prone to further error.

Legacy digital formats. A routine examination of format lists in standard file identification software, online format registries, and preservation-specific registries indicates the existence of thousands of file formats, many of which are no longer in common use. Identification and verification of legacy formats is a difficult problem due to gaps in format registries, ambiguous or conflicting signatures associated with various format types, errors or inconsistencies in changing versions of specific formats over time, proprietary and undocumented formats and format features, and problems associated with technologies such as digital rights management and compression and encryption schemas. Digital preservation technologies must provide reliable, high-performance format identification and verification mechanisms in order to handle large, heterogeneous collections.

Archival handling. Avoiding loss and lossy alteration of digital materials is an essential goal of long-term digital preservation. Yet the risks to digital content are often radically different than those associated with traditional analog media. Digital objects can be subject to numerous and subtle forms of alteration and information loss due to maintenance and access activities, risks as significant as those associated with simple neglect. Archives typically mitigate such effects by maintaining original bitstreams in secure storage, associating materials with high quality metadata, and committing to interoperability standards in order to share and verify holdings.

Digital object migration and access. Legacy digital documents and media, particularly those in proprietary or poorly documented formats, can be difficult and expensive for archives to maintain. Modern software environments may not render them properly (or at all) and retaining both the

software and knowledge required to access them can increase both cost and risk. Digital object migration, or the interpretation and transformation of legacy documents into modern, open formats, is a widely accepted technique to address this problem. However, the migration process itself is problematic: ensuring that all required properties of the original document are retained is a complex task that is frequently at odds with the desired level of automation. Furthermore, document analysis techniques that may be appropriate for some formats do not typically apply to others. Digital object migration remains an area of intensive research.

Context and emulation. Digital content is often designed with significant assumptions about the environment in which it will be rendered or otherwise manipulated. While these assumptions can seem self-evident – dependence on a physical access or input device, a particular version of Windows, a specific release of an office document suite – as digital materials age almost any assumption begins to present a serious impediment to access. This is commonly exacerbated by a lack of sufficient documentation; however, documentation or provisions for extended metadata do not address the fundamental problem of ensuring simple, reliable access to the data. Emulating the original environment through the use of a virtual machine comprised of a legacy operating system, supporting software, and virtual device drivers, provides part of the solution to this problem. Providing a path between the contextual information required to reconstruct the required environment and the automation of a VM for access is further addressed in the work presented here.

Preserving Government Documents

In the United States, the Federal Depository Library Program incorporates almost 1,250 depository libraries as conduits for access to government documents by citizens around the nation. Among these holdings are approximately 5,000 CD-ROMs published by the Government Printing Office beginning in 1987 and distributed for local access via dedicated workstations within the

libraries [39]. In more recent years, distribution of digital materials produced by government agencies has increasingly shifted to the web. Most of the data distributed on these CD-ROMs, however, remains offline. While depository libraries are tasked by the Depository Library Council with “organizing systems for transparent, cost effective collaborations to provide services and resources to end-users and colleagues”, many lack the technical expertise and funds to build online collections from these aging sources [34, 25].

There are numerous reasons to engage in the production of online collections. In their current form, the GPO CD-ROMs can only be accessed on-site via dedicated machines which may or may not provide the legacy software environments needed to view all of the materials. Online access not only eliminates the need for physical presence, but provides many technical and administrative benefits. Online collections can be sourced using a distributed model, in the case of the FDLR CD-ROMs allowing institutions with disparate holdings to contribute disk images to a shared and ultimately comprehensive collection. Multiple copies of individual items can be compared to provide “gold standard” reference materials (something that cannot be achieved in an individual collection). All metadata associated with the collections – administrative, descriptive, and preservation – can be stored alongside the online collection and used not only for search and maintenance but also to crosswalk collections distributed between multiple institutions.

1.2 Approach and Contributions

In this work I show that high-quality, low-cost virtual collections can be created from large, heterogeneous sets of digital objects existing in filesystems extracted from removable media. I demonstrate that most *technical* risk to these types of “born-digital” collections can be effectively managed through the use of mature open source software tools and libraries along with custom

software and scripting to process metadata, control migration events, interact with and manipulate emulation environments, and provide a web-accessible interface to the materials. These claims are supported by experiments on millions of media-rich documents spanning hundreds of digital formats held in the virtualized FDLP CD-ROM collection at Indiana University – materials which are publicly available on the web and directly via a distributed, networked filesystem.

A core contribution of this work is in demonstrating that preservation-friendly collections can be constructed using available technologies without sacrificing access. In order to do this, I incorporate methods into a model which avoids single points of failure which could lead to information loss. In addition to maintaining bitstreams from original sources, I use highly reliable but easily replicated methods to provide migrated renditions of documents where possible, and user-oriented emulation mechanisms for emulation where necessary. These techniques are treated as complementary, and are incorporated into a file organization and access model that reduces the level of technical expertise required by end users wherever possible. The model is designed to be sustainable, modular, and interoperable with existing and planned archival systems, principles that facilitate the development of larger infrastructures into which such collections are often incorporated [58].

Based on data collected and analyzed during processing of the FDLP CD-ROM collection, I present novel insights into lowering risk during imaging of removable media and outline methods that can be easily replicated in other collections. I examine why it is important to retain *filesystems* rather than simply extracting collections of digital objects, and provide a logical schema for organizing both data sources and associated legacy metadata in a manner than supports simplified navigation and indexing. I show that *in most cases*, automation of open source document and media processing platforms can be used to provide reliable access to legacy digital objects over the web without introducing additional risk or requiring alteration of the original bitstream. I examine

strategies to retain the *contextual knowledge* needed to install and use many legacy executables, and show how this can be integrated in a system for the automatic reconstruction of legacy computing environments within virtual machines.

Preservation Issues and the State of the Art

Fundamental scientific, methodological, and technical issues can be identified in existing preservation techniques. Many of the tools and frameworks currently in use by the digital preservation community remain under heavy development, a consequence of extensive planning and lifecycle considerations across all aspects of data handling, including creation, acquisition, ingest, metadata construction and handling, access, and storage.

Yet to date, relatively sparse *empirical* data has been published on in-practice successes and failures in identifying, maintaining, transforming, and supporting continued access to legacy digital materials. In part this is because many digital preservation efforts are nascent, and large (particularly national and international) repositories have only recently – within the past decade – completed implementations of long-term preservation plans that satisfy necessary technical, administrative, and collaborative goals. Here I identify some of the open issues addressed in contributions of this work, and examine related research from within the preservation community.

Few organizations have published technical information on internal processes for preservation handling of removable media beyond care and handling [9]. Unfortunately, legacy removable media exhibit a number of significant risk factors beyond simple physical decay. While CD-ROMs are governed by a standard (ISO 9660) that protects data using 3-layer error correction, access to data preserved on these media can be compromised by lack of conformance to the standard during creation or inappropriate contextual dependencies embedded in the data itself. For example, the “resource” and “data” forks found on legacy Macintosh-authored CD-ROMs cannot be read

properly on a Linux platform without modification of the appropriate libraries. A variety of extensions have also been appended to the ISO 9660 specification over time, resulting in varying levels of compliance observed within large collections [115]. A number of publications exist detailing the production and inventory of the FDLP materials [12, 27, 49] but do not address these filesystem issues. A primary contribution of this work is the identification and evaluation of reliable, repeatable techniques to extract CD-ROM filesystems without information loss in order to support future access.

The digital preservation community has focused significant effort on the development of preservation specific registries to record detailed information about legacy file formats (including structural characteristics, known versions, known or accepted migration paths) along with software tools to process collections in order to extract this information [7, 90, 35, 33, 24, 51]. Currently, the majority of these projects lack the performance and scope to support sensible interactive and access environments. In this work I demonstrate the application of widely-used open source file identification tools to support on-demand access to hundreds of legacy digital formats.

Provisions for access must account for the diverse software environments originally associated with the file objects themselves – environments which may have been dynamic, supported by commercial software, or incompatible with traditional archive arrangements by design. For example, archived HTML files often contain absolute links that do not function properly outside of their original server context. Media inclusions, document metadata, and other features can be lost when importing legacy Microsoft Office documents into OpenOffice, and the project provides an extensive migration guide documenting additional issues.¹ While there exist published accounts of migration in working repositories [10], relatively few publications provide sufficient detail to replicate the strategies used.

¹<http://documentation.openoffice.org/manuals/oooauthors2/0600MG-MigrationGuide.pdf>

In an emulation context, many custom binaries make specific demands on how hardware devices are configured in order to be executed. Much of the existing research on emulation by digital preservation experts has concentrated on its relative merits versus migration [98, 97], the degree to which it captures the desired properties of rich or interactive media [68], and preservation of the emulator itself [108, 109]. Most of these explorations do not address the problem of context – that is, the information required for users to install and interact with legacy content.

1.3 A Preservation Testbed for Removable Media Collections

In the following chapters I describe work carried out during the past three years with Dr. Geoffrey Brown at Indiana University to produce a viable online repository of government documents from the FDLP CD-ROMs. This work describes media analysis, transfer, and media image virtualization of the Federal Depository Library Program CD-ROM archives, identification of common risks associated with legacy file format features during migration or rendering in modern software environments, automated access to targeted emulation environments, and the transfer and manipulation of existing metadata through the use of modern extensible XML schemas.

I demonstrate the feasibility of creating a web-accessible archive from these legacy materials that maintains original bitstreams, mitigates risks associated with the original physical media collection, and improves the capabilities of the holding institution with regards to collection verification and distribution. This approach is designed to be “preservation-aware” from the ground up; it does not constitute a formal digital preservation strategy, but the novel techniques introduced here can be readily introduced into existing digital preservation systems. Significantly, this work directly serves the practical needs of libraries wishing to take aging and sparsely used collections of legacy digital media and improve community access to the materials they contain by making them

available on the web. Most institutions have limited resources to reprocess such collections; the emphasis on low-risk automation and use of commonly available open source technologies here makes the creation of high-quality virtualized collections more viable. Figure 1.1 provides a broad overview of some of the technologies used for distribution and access in this work.

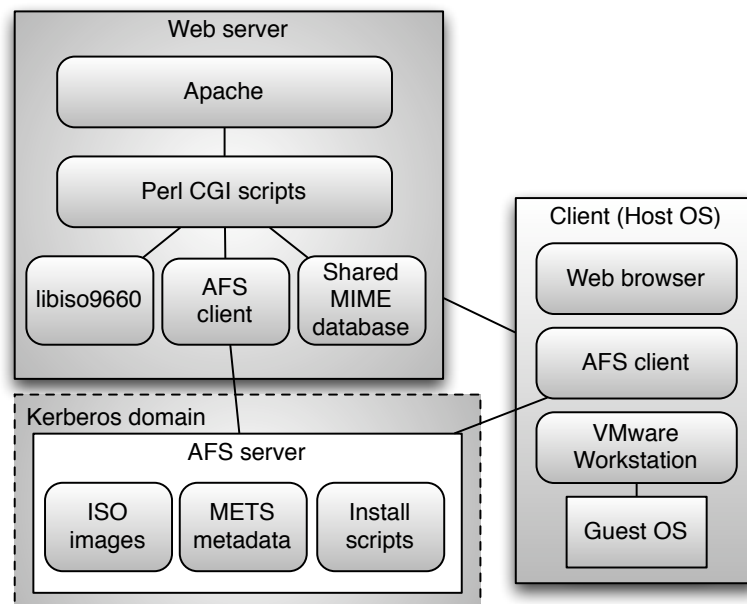


Figure 1.1: Web service for distributed access to legacy documents.

For collection processing, I study strategies to improve file-format identification which combine existing open-source tools with scripted tests and heuristics, providing breadth of coverage while tolerating failures gracefully. I show that widely-available open source format identification tools such as `libsharedmime` (based on the Shared Mime-Info Database specification) can be used to provide high-quality access to heterogeneous collections. This particular implementation has a number of advantages over other available file format registries [90], [35]. It is production quality, fast, integrated into the Unix environment used here for migration services, has an easily

customizable database, and produces succinct machine-readable descriptions of identifications.

I have constructed reference implementations to demonstrate the feasibility of using off-the-shelf tools to assist in semi-automated dynamic and batch-mode file migration and web-based file access to legacy materials. These include experiments in automation of OpenOffice using the Python API and a server daemon to construct a multithreaded migration environment that is fast (< 6 seconds per document on average) and failure-tolerant. An alternative uses lightweight scripted access to the Gnumeric `convert` utility to generate on-request migrations for legacy spreadsheet formats.

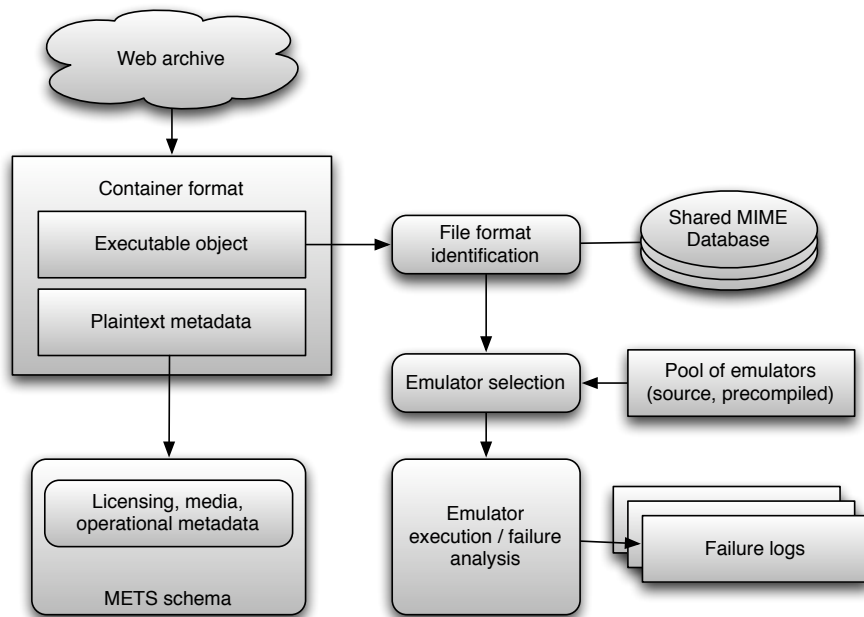


Figure 1.2: Analyzing rendering performance in an emulated environment.

Migration, Emulation, and Rendering

A legacy digital object selected from an archive will typically be rendered in one of several ways. In an emulation scenario, the document is rendered using a legacy software tool in an appropriate operating environment. The document may be directly rendered using a compatible modern software package (for example, opening a Microsoft Word 97 document in OpenOffice 2.0). Finally, a migrated object may be generated from the original binary.

There are risks associated with each of these scenarios. Emulation of a legacy environment requires selection and configuration of an operating system, software packages, and supporting libraries. Correct rendering of a legacy binary directly in a modern tool depends on knowledge of the format structure and reimplementing of software to access that structure through published documentation or reverse engineering. For example, the OpenOffice consortium maintains a list of incompatibilities with legacy Microsoft Word documents, and notes the existence of undocumented features in the Word format. Full migration of a given document must account for relevant features of the originating format, mapping those features to the destination format, and compliance with the structure of the destination format.

Providing high-quality renditions of legacy binaries in modern formats does not always require adherence to the composition of the original document. A simple columnar single-worksheet spreadsheet composed in Lotus 1-2-3 may be adequately exported as a Comma Separated Value file, or viewed as an HTML page with limited formatting. We may tolerate font substitution where legacy bitmap system fonts have been superseded by TrueType equivalents. For many basic access scenarios, the tools we use should be able to reliably identify document type, match it to a modern format and/or environment, and log failures for future improvement. I demonstrate ways in which this process can be automated for key document types using a daemonized server to feed incoming migration requests to individual client processes, as shown in Figure 1.3.

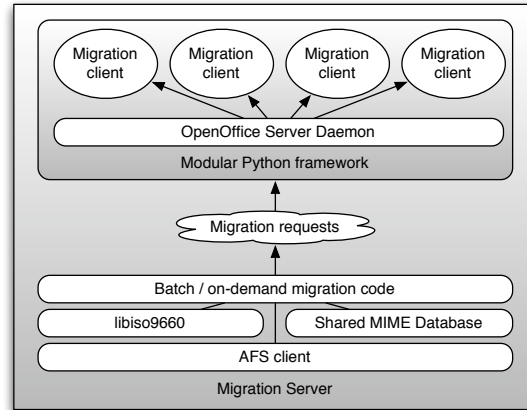


Figure 1.3: Automated document migration with OpenOffice.

Document migration decisions in archival workflows are often driven by administrative controls that do not adequately address the risk of information loss. For example, many archives “normalize” incoming word processing documents to the Adobe Portable Document Format (or the long-term preservation equivalent, ISO 19005-1 compliant PDF/1A-a/b). Most publications describing this process discuss the normalization procedure only tangentially, and rarely explicitly state which tools are used. This is a serious concern, as many translation tools – including export directly from originating commercial packages such as Microsoft Word – may produce formatting errors or loss of information.

Normalization to the archival PDF standard is a particularly problematic example. The formal description specifies that fonts must be embedded (and legally embedded) for universal formatting.² However, font identification in legacy word processing documents remains a difficult problem. Microsoft Word documents rarely contain embedded fonts [112], and silently fail on fonts that have licensing restrictions. Matching font names extracted from the document structure to system

²<http://www.digitalpreservation.gov/formats/fdd/fdd000125.shtml>

fonts (which have been altered across the various iterations of Microsoft Windows) and third-party font databases is complex due to variations in name conventions. Additionally, the archival PDF standard specifies that no audiovisual content may be included, excluding a small but significant percentage of Word documents from this particular migration procedure.

Similar concerns exist for other office documents. Consider the sizable collection of legacy Lotus 1-2-3 spreadsheet files, support for which has recently been discontinued in the Microsoft Office suite. In previous releases of Microsoft Excel, a host of issues can arise in attempted conversion of Lotus 1-2-3 files. Macro buttons, text boxes, and any drawing elements are not converted. A special ‘Transition Formula Evaluation’ mode must be used to ensure proper handling of boolean formulas or formula entries containing both text and numerical data. Order of precedence differs between Microsoft Excel and Lotus 1-2-3, and links to external resources in a Lotus 1-2-3 worksheet must be manually restored on opening the document. Prior to Excel 97, dates post-2078 produce two *separate* errors depending on the date function used.

In spite of these problems, existing publications indicate that many large, well-funded archives (including Federally-funded institutions) perform ingest procedures that do not adequately address these risks. In many cases, the original bitstream is retained, enabling future archivists to re-examine the problem. In others, it is not. The Dark Archive In The Sunshine State, an NDIIPP-funded project [14] holding terabytes of government-produced data, normalizes documents to PDF on ingest and discards the bitstream due to lack of available space.

These examples of risk form part of a larger concern in the preservation community, that of *durable encoding*. This is the idea that long-term preservation is best served by a minimal set of durable, transparent, open standards, such as unicode character encoding and a subset of the XML standard. In principle, this could assist in reducing the need for “transformative” migration procedures, in which the semantics or application behavior associated with particular document formats

is typically not maintained. In practice, transformative migration is justifiable for many access scenarios in which the risk of information loss is low and the requirement for environmental authenticity is negligible. In particular, I have shown that such procedures may be advantageous where transfer time over a network is a consideration, or users require only partial renditions to locate the required information [114].

A major problem with rendering documents in existing archival products is the “coarseness” of the approach used in migration of legacy digital objects to modern representations. I discuss future directions for establishing methods to improve rendering through the selection and implementation of techniques to compare document composition in migrated objects with that observed in a legacy environment and/or in exported renditions produced by the originating software package.

1.4 Chapter Outline

Chapter 1 introduces the thesis, describing the general importance of long-term digital preservation, the need to accelerate research efforts in order to preserve critical materials, and the ways in which this field intersects with computer science. It concludes with this outline.

Chapter 2 examines existing technical approaches and methodologies developed by the digital preservation community. I focus primarily on experimental studies and software tools related to the handling of aging formats or those for which documentation is incomplete. Specific examples of metadata schemas, preservation workflows, and archival packaging of digital materials are included to provide a clear framework for the more focused issues discussed elsewhere. I briefly examine limitations uncovered by or inherent to existing approaches.

Chapter 3 presents a detailed analysis of the outstanding technical issues, specifically with respect to extraction of bit-accurate images from legacy media, file-format identification, interpretation of binary-encoded digital objects, and feature extraction. This includes technical background focused on bit-stream handling and adherence to common format standards. Additionally, a complete description of experimental design is provided.

Chapter 4 describes work carried out to process, analyze, and develop an access platform for a collection of born-digital materials drawn from a large legacy media collection. I examine costs associated with analysis and processing of a legacy collection, describe specific methods by which existing metadata sources can be automatically and reliably transferred to preservation-friendly schemas, and discuss the underlying design of the “virtual filesystem” used to support flexible storage and access methods.

Chapter 5 addresses experimental work carried out to provide both bulk migration and *migration on demand* for a range of document and media formats extant within the collection. I provide additional background on existing migration research, describe high-quality (and low cost) methods to perform migration using open source tools and document processing suites, and analysis of time and space requirements for the migrated documents.

Chapter 6 examines the development of software providing automatically customized emulation environments to support access to legacy binary documents using contemporaneous operating systems and supporting software. I discuss the use of custom scripting, metadata, and web-interactive tools for off-site access.

Chapter 7 discusses the contribution provided by this work and describes future directions for the research. These include mechanisms to support risk reduction in common archival practices, quality metrics for preserved documents, and the use of distributed storage systems to enable transfer, verification, and reconstruction of legacy, possibly damaged, media from multiple sources.

Appendix A addresses further issues related to identifying and mitigating risk in digital collections, and discusses methods to simplify selection and presentation to human evaluators of “high-risk” documents for targeted document features. A case study on font analysis of legacy Microsoft Word documents is presented.

Appendix B provides technical detail and code samples for ISO 9660 filesystem verification, automation of open source software for digital object migration, and a demonstration of an efficient customization and control wrapper for VMware-based virtual machines.

Digital Preservation Overview

2.1 Introduction

Ensuring that digital materials of historic, cultural, scientific, and economic interest will remain accessible in the future implies a commitment not only to developing, evaluating, and applying the required technologies, but also to an ongoing process of management and curation in order to guarantee the viability of these objects under consistently shifting access conditions. The inherent structure and management of the preservation process guides many aspects of related research, which focus on identifying specific needs and risks associated with digital collections and developing methods to reduce both vectors for information loss and costs – particularly those involving human labor.

Discussion of our evolving understanding of preserving digital materials is not limited to the academic literature. In a popular article entitled “The Myth of the 100 Year CD-ROM” published in 2004 in the United Kingdom newspaper *The Independent*, the author notes the now commonly accepted fact that recordable CD-ROMs stored in non-ideal conditions can degrade far more rapidly

than manufacturer tolerances would suggest.¹ The author indicates that data recovery experts and archivists (at the time of writing) rely on multiple backups or older but more robust magnetic tape media.

What is not addressed in the article is the fact that even if one were to discover a cache of viable CD-ROMs 100 years from now, it is unlikely that the technologies required to recover and interpret the bitstreams contained on them would be readily available. Over the past decade, digital preservation research has focused on many of the issues surrounding the implied problem: simply creating a “vault” of digital materials on high-quality media fails to capture why we might preserve digital materials in the first place – in order that they may be accessed and used by future generations.

Defining Digital Preservation

In 2002, the Research Libraries Group (Online Computer Library Center) commissioned a report in which they examined the challenges and the state of the art of digital preservation.

Digital preservation is defined as the managed activities necessary: 1) For the long term maintenance of a byte stream (including metadata) sufficient to reproduce a suitable facsimile of the original document and 2) For the continued accessibility of the document contents through time and changing technology.

Constructing high-quality archives of born-digital materials that address endemic risk and control costs while maintaining (or adhering to) a mandate for continued access is a complex and – increasingly – explicitly managed task. The transfer of data from legacy media to modern storage systems, migration of legacy objects to open and preservation-friendly or preservation-aware formats, and emulation of legacy operating systems and environment all present numerous technical

¹<http://www.rense.com/general52/themythofthe100year.htm>

and policy challenges. In addition, tracking provenance and ensuring that any associated metadata is maintained or migrated to modern preservation-specific standards adds further complexity.

With the inexorable shift towards ubiquitous networked and distributed storage systems, many of the most pervasive problems in preserving digital information become increasingly abstracted away from the basic storage hardware. That is, while indefinite bitstream preservation is increasingly feasible, the contextual and access problems associated with objects encoded in those bitstreams remain.

In this chapter I review some of the fundamental methodologies, technologies, and frameworks that form a basis for long-term digital preservation initiatives. These include the digital object models that are used to describe the roles and functional responsibilities of various entities within a formal digital archive, core technical issues in bit preservation, metadata construction and handling, identifying and verifying digital formats, and finally migration and emulation.

2.2 Digital Object Models

Reference models for archival handling of digital objects form a cornerstone of most modern digital archiving systems. In general, they are not blueprints for implementation but rather outline administrative, technical, and organizational roles, relationships, and activities that occur within the archive. They allow archives to evaluate activities and outcomes within a common context, and provide a reference for incremental improvements as additional requirements are identified in practice.

Reference Model for an Open Archival Information System

The Open Archival Information Systems digital object model outlines the role of digital objects within a formal archival specification. Development of this model was originally led by the Consultative Committee for Space Data Systems (CCSDS) [32], and has subsequently been adopted as the international standard ISO 14721:2003. Every data object that has, at some point, a physical manifestation on some media. That data object is part of an information object (through association, and given some formal representational language) which itself is explicitly packaged in a wrapper that makes it useful in an archival context.

This object model forms the theoretical foundation for a wide variety of archival activities. A particular digital object – having been retrieved from a non-archival source or newly authored – may be submitted to an archive as a Submission Information Package. The delivered package is verified and stored per the mappings of the model onto a particular architecture (that is, an archival repository or software framework), subject to internal maintenance over time, and disseminated to users as required. All stakeholders involved in this process are referred to as the ‘Designated Community’, a term that includes *users* as well as producers and those who serve as direct mediators of the archive.

In Figure 2.1, the Digital Object exists in a particular instance on some physical media (Physical Object). Together these form a data object which, together with a Knowledge Base consisting of information required to understand the information being archived or received and Representation Information pertaining to how the object is to be rendered or interpreted, form an Information Object. Information Objects are characterized in classes at a high level as Content Information, Package Description, Packaging Information, or Preservation Description information (PDI). The PDI associated with a particular instance of Content Information to form an Information Package is

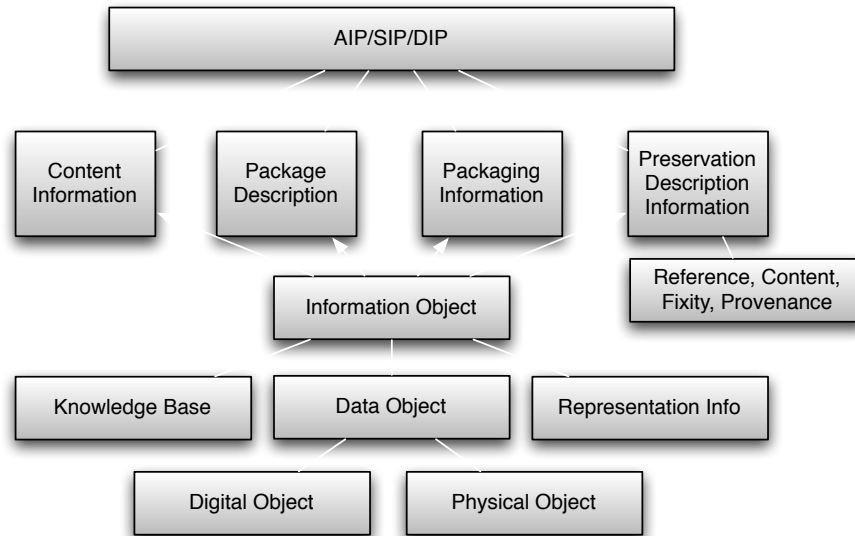


Figure 2.1: OAIS Document Object Model

itself defined in terms of supporting four roles. *Fixity*, or incorporation of data integrity checks, *Reference*, or support for identification and location over time, *Context*, or establishing how an instance of Content Information relates to the supporting environment, and *Provenance*, or documentation of past actions on the Content Information.

In practice, OAIS-compliant repositories are governed by a functional model (Figure 2.2 that defines, as the name implies, functions that are addressed in the receipt, handling, and distribution of archival materials [32]. At the front end these include ingest services and functions that accept Submission Information Packages from producers, along with functions to prepare Archival Information Packages for storage. Archival Storage services and functions explicitly support the storage and retrieval of AIPs, including media updates, error testing, backup, and disaster recovery. Alternately, Data Management functions are responsible for maintenance of and access to descriptive

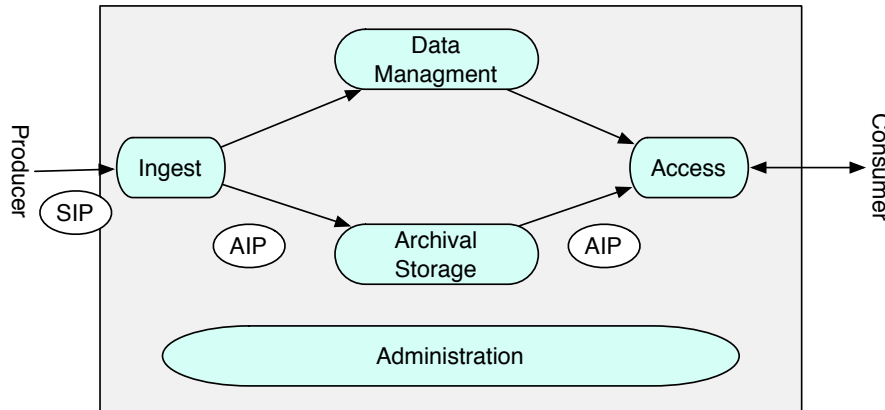


Figure 2.2: OAIS Functional Model

and system information. Basic implementations may use a database along with supporting scripts and configuration files; in large repositories with extensive capabilities, data management can be extremely complex [37].

It is important to stress that OAIS is not an architectural model; it has no direct mapping to a particular implementation and as such does not guarantee interoperability between any set of implementations. However, it is both sophisticated and modular. Existing implementations often select parts to omit; for example, certain dark archives discard the access requirements. Others specify particular administrative tasks that are not incorporated.

Fedora, or the Flexible Extensible Digital Object Repository Architecture, maps ingest and dissemination functions from OAIS. *Fedora* uses an internal content packaging schema called *Fedora Object XML*; widely-supported and open source, *Fedora* is used for a large variety of services including search, preservation metadata handling, administrative tasks, and the use of modular application add-ons designed by the community. Another major repository systems which implement parts of OAIS include *DSpace*, an institutional repository originally designed for research output

(and which uses METS metadata internally). In 2009, the organizations charged with maintenance and community involvement with these projects combined resources to form “DuraSpace”, an effort which notably has begun development of a cloud-based hosting service to improve sharing and reuse of existing content.²

2.3 Bit Preservation and Removable Media

The inherent problems in maintaining bit-accurate copies of files and filesystems on removable media are well understood. Removable media are generally fragile, suffer increased degradation and usage increases, and place significant overhead on institutions tasked with maintaining them. As a consequence, many libraries and archives attempt to follow “best practices” guidelines for handling and physical preservation of these materials as access conditions allow.³ Such practices are, at best, simply bandages that mask the underlying problem.

Problems with maintaining infrastructure support for materials access have driven acceleration of efforts to transfer legacy digital materials on removable media to more stable platforms. Most institutions recognize that while well-preserved CD-ROMs may, under ideal conditions, have shelf-lives measured in many decades, the hardware, operating systems, and original software installations required to access them will long since have vanished. A “museum” approach to maintaining this hardware in perpetuity is inordinately expensive and extremely risky, and is generally not considered to be a long-term preservation strategy except in cases where such software may depend on custom built or one-off hardware platforms.

As the cost of both consumer and Enterprise-grade storage systems falls while bit-densities increase, the more significant issue is ensuring that bit-level preservation is in fact being performed

²<http://www.librarytechnology.org/ltg-displaytext.pl?RC=13958>

³<http://www.aallnet.org/sis/tssis/tssl/22-04/presrv.htm>

reliably. Typically, archival storage utilizes backups (both local and remote) and checksums to ensure data integrity. Systems such as Lots of Copies Keep Stuff Safe have been developed over the past decade to provide additional measures against intrusion and disaster [62]. In related work, David Rosenthal has examined the claimed versus measurable reliability of modern disk systems given a hypothetical measure of bit damage. Additional work on trusted archiving, maintenance of digital signatures, and inter-system trust can be found elsewhere [41]. Additional studies have further examined the scalability of digital archives built on common open architectures such as DSpace, Fedora, EPrints, and Greenstone [2, 56, 113, 6].

2.4 Metadata

Virtually all forms of digital content are distributed in association with metadata – that is, data to identify, describe, and support the raw content. In an archival context, metadata typically supports procedures that identify and define specific objects and filesystems, provide information on the structure of particular content, and delineate the administrative activities associated with that content. In this context, metadata can support complex goals such as ensuring authenticity, provenance, and interoperability between platforms. Typically, implementations associated with various metadata standards are associated with XML schemas designed to facilitate long-term preservation without introducing significant format or structural risk. These standards vary in complexity; several are described below.

Metadata Standards

The Dublin Core element set (ISO Standard 15836) provides a small element set consisting of 15 fields designed to support description of physical, digital, and networked (web page) resources. A

Qualified version of the element set exists to support the additional roles of Audience, Provenance, and RightsHolder. The Dublin Core set is flexible enough to work across many domains, and as a consequence is used in many library and academic applications. Dublin Core is a relatively mature standard, and is incorporated into a number of repositories including DSpace (in Qualified form).

METS, or the Metadata Encoding and Transmission Standard, is an XML-based schema designed to describe administrative, descriptive, and structural metadata, with an ontology specifically designed to capture multilayered structures in digital repositories. METS documents are sectioned to include a header, descriptive metadata, administrative metadata, a file section, a structural map, structural links, and behavioral information for executable content. METS – as with certain other schemas – is designed to address nuances of digital libraries that do not generally occur when dealing with physical documents. Specifically, a properly constructed METS record allows digital documents to be identified, evaluated, and maintained in ways that capture not only the features of the raw bitstream, but also the methods of production, associated access requirements, and relation to other digital holdings. METS records can form the basis for “packaging” of archival data for ingest, preservation, and dissemination to external entities [66, 65].

The MARC, or MACHine-Readable Cataloging standards, are used predominantly to represent and disseminate bibliographic information typically associated with library holdings. MARC records are composed of a record structure, designation of content, and associated data content. At the time of writing, the MARC standards and associated format designs date back nearly five decades; the specificity with which these formats cover existing materials and widespread use (billions of associated documents) mean that while outdated, MARC continues as a common standard. An associated XMLized schema, MARCXML, is based on one variant of MARC, and is commonly used to make bibliographic data web-accessible [63].

MODS, or the Metadata Object Description Schema, represents a waypoint between the complex but outdated MARC formats and simple but coarse schemas such as METS. Particularly for born-digital materials, MODS can serve as an appropriate technical bridge for incorporating bibliographic metadata from existing library systems into the modular METS schema in order to build metadata records that can be incorporated into fully-fledged preservation systems [75].

Preservation Metadata

As libraries and archives have amassed larger collections of born-digital materials, preservation researchers and standards agencies have recognized an increased need for metadata architectures that are “preservation aware” from the ground up. In response to this, the Preservation Metadata: Implementation Strategies (PREMIS) Working Group has created a “data dictionary” for preservation metadata [89]. The goal of this work is ambitious: to make archived digital objects self-describing over long periods of time, ensuring that the access and rendering technologies, rights management issues, and administrative tasks associated with those materials are documented and consistent as archival architectures are maintained and grow over time.

The model is agnostic with respect to implementation, although an XML schema has been produced for support in digital archives. The data model emphasizes core entities that embody typical preservation activities, including but not limited to entities associated with preserved objects, rights management, and activities performed on the object over its lifetime.

The PREMIS model attempts to formalize some problematic aspects of variegated digital materials that are not well addressed in existing metadata schemas. First, the notion of *inhibitors*: any features of a digital object that may prevent or provide barriers to migration or future access and use. These may include password protection, encryption, or other mechanisms based on key management [11]. PREMIS also provides basic event types through which *provenance* information

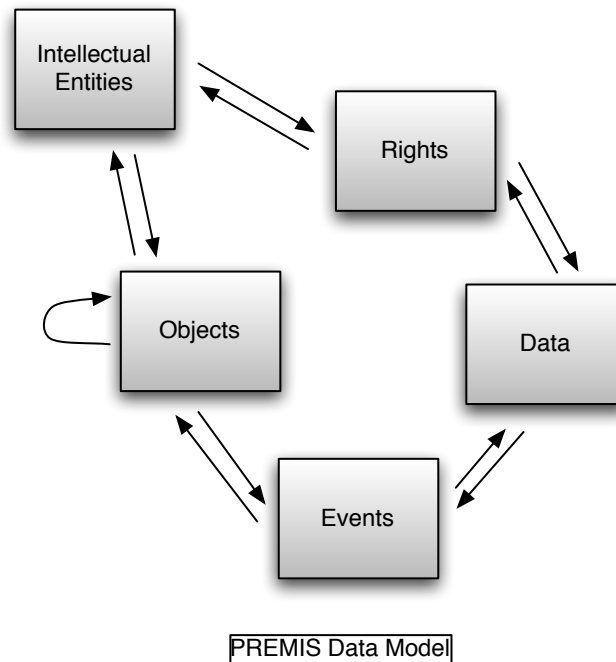


Figure 2.3: PREMIS Data Model

can be encoded, documenting a production and changelog history for a particular digital object. Beyond simple preservation of a bitstream, digital objects may be described in terms of *significant properties*, or properties of an object that are essential to its proper reproduction and interpretation in the future. These may include fonts, color maps and image resolutions, and other factors affecting the “look and feel” of a document. The semantic units defined by PREMIS allow for flexible encoding of this knowledge based on mappings from external schema. Finally, PREMIS allows archives to record rights metadata to encode legal restrictions and permissions information to support a variety of access models.

2.5 Format Identification and Verification

Digital format technologies are constantly being introduced, updated, and – in many cases – retired. This presents a serious technical issue for repositories tasked with maintaining access to large and heterogeneous collections. Not only must they be able to automatically and accurately identify and validate objects in different formats, but they must take specific actions to ensure that supporting technologies are also maintained; when the objects can be migrated without loss, they must be able to make justifiable choices for preservation formats based on information held in a variety of registries. Within the past decade, a number of preservation-specific format registries and associated identification and verification tools have been developed. Some of the most prominent are described in this section.

PRONOM is a technical registry designed from the ground up to support long-term preservation. Designed and maintained by the UK National Archives, an online database can be used to search all current format information in the registry.⁴ *PRONOM* includes the DROID, or Digital Record Object Identification tool to assist in performing batch file format identification jobs. Early versions of DROID suffered from limited coverage due to the small size of the registry. This has improved over time, although the DROID tool must still be used in coordination with other software to provide full coverage [7]. Significantly, *PRONOM* includes a facility for persistent, unambiguous identification of records within the registry, known as the “PUID” or *PRONOM* Unique Identifier scheme. In the long term, the utility of this scheme will depend on the growth and acceptance of *PRONOM* in practicing institutions. To date, the penetration of this tool in production settings is limited outside of certain European institutions.

JHOVE, or the JSTOR/Harvard Object Validation Environment, is an extensible framework for

⁴<http://www.nationalarchives.gov.uk/PRONOM/Default.aspx>

format validation. At the time of writing, the latest version of the associated tool (also known as JHOVE) is 1.3, although a complete rework of the architecture has begun with JHOVE2. JHOVE attempts to analyze formats for both *well-formedness* and *validity*, typically relying on byte sequences, type-specific significant features, and other information on formats to ensure reliable identification and confirmation of an object. As with some of the other tools examined in our research, JHOVE currently exhibits limited coverage, lack of persistent analysis, and significantly worse performance than equivalent open source C tools generally used for file identification in operating systems. JHOVE does, however, provide a relatively sophisticated API and XMLized output of file validation scans.

More recently, PRONOM is being integrated with a nascent program also hosted at Harvard – the Global Digital Formats Registry – to attempt construction of a Unified Digital Formats Registry (UDFR) which leverages technical information used in PRONOM along with distributional modifications to allow localization of the database, and increase of community involvement in directional concerns for the project. Versions of the PRONOM registry are also used extensively in commercial products.

2.6 Migration and Emulation

Urgent calls to address the multitude of issues associated with software obsolescence have appeared over the past decade [43, 96]. Two essential strategies have emerged as potential solutions to this crisis: the creation of new software systems that enable conversion of obsoleted formats (migration) to modern renditions, and the execution of legacy operating environments and software (emulation) to provide access to binary executables and proprietary documents [23, 78].

Migration

“Migration on request”, or migrating a document from the original bitstream using the most current tools either periodically or on each access event, is a high-level form of risk reduction. The accuracy and scope of migration tools improves with time and additional development effort. Maintenance of the original bitstream is therefore generally recommended to avoid compounding error or information loss by migrating from non-original sources over time [67].

Relatively few directed technical studies of risk in handling various formats exist. One of the earliest is a study by a Cornell University research group that identified a limited set of risks in conversion of statistical functions from Lotus 1-2-3 [57]. The study performed simple conversions using two distinct commercial software packages (both of which have subsequently been obsoleted) and prepared a manual analysis of the results. The study concluded that no off-the-shelf commercial packages could be recommended for document conversion. The CENDI Digital Preservation Task Group conducted a survey in 2007 on a small number of formats for government document preservation [45]. A general discussion of preferred characteristics for preservation formats for the Wellcome Library is given by Thompson in [104].

Lars Clausen has extended ideas introduced by Rothenberg, the PLANETS project [88], and Rauber and Rauch [91] to construct a method for semi-automatic quality assurance in document transformation. Clausen describes the use of a generalized file format abstraction framework to store key “aspects” of a document type, in order to better direct the construction and application of tools to extract and analyze relevant features. Clausen provides an explicit measure for *normalized quality* of a document as $\frac{Q_m(o) - \max(0, E_m - \sigma_m)}{2\sigma_m}$, where $Q_m(o)$ is the quality, or inverse error, for a particular property m of an object o , with average and standard deviations of the measure are, respectively E_m and σ_m . This metric is somewhat limited, as it takes a simple average of all normalized values for any significant properties considered. Clausen’s proof-of-concept work includes a

small set of Microsoft Word documents exported to PDF from both Word and OpenOffice 2.0, with the resulting document sets compared for basic measures such as page length, font substitution (using the `pdffonts` utility to extract font names) and number of words [16].

A selection of other studies examine risk to legacy documents using both modeling and human analysis. McCullough et al. have performed a brief study on statistical errors prevalent in multiple version of Microsoft Excel [64]. The INFORM methodology devised at the Online Computer Library Center describes a model incorporating risk categories, specific risk factors associated with each category, and scales to describe both probability of occurrence and impact [102]. The scales incorporate organizational issues and other factors external to features of the format, with risk-assessment data collected as a group-consensus average of individual manual assessments. Finally, there is ongoing work in format obsolescence detection and notification within archives [84, 20].

Emulation

Emulation of obsolescent hardware environments is another strategy used to preserve long-term digital preservation. Emulators replicate the functions, behavior, and – when combined with appropriate legacy software environments – the “look and feel” of legacy computing platforms on modern workstations. In spite of the relative costs associated with their creation, the use of emulators is considered a viable preservation strategy for several reasons. First, when used as part of a formal digital preservation plan, an emulator can help in avoiding information loss due to ongoing migration procedures. As long as the emulator itself remains viable, storing the original bitstream, along with an operating system and supporting software in a virtual machine, provides a guarantee that we will not lose our ability to render the object. Second, for those digital documents and media for which environmental authenticity is a concern – particularly interactive content, executables which depend on “seeing” a particular kind of hardware, or documents for which

there is simply no known migration path – emulators provide a consistent operating environment.

Much of the work on emulation in the preservation community has focused on questions that largely ignore the requirements of the end user. These include “Will a reliance on emulation adversely affect other preservation strategies?”, “How can we ensure that the emulator itself is preserved?”, and “What vulnerabilities are exposed when relying on imperfect emulation methods?”. We believe that the issue of customization on demand – that is, whether we can reliably provide users with emulated environments to suit their needs on a per-resource basis, is of equal importance. This is not just a technical issue, but one of building a sensible access requirements for existing and future archives.

In addressing the issues, the OAIS model makes specific reference to legacy Windows operating systems, and notes the following:

[D]etermining that some new device is still presenting the information correctly is problematical and suggests the need to have made a separate recording of the information presentation to use for validation. Once emulation has been adopted, the resulting system is particularly vulnerable to previously unknown software errors ...[32]

The suggestion that a “reference recording” be retained to verify software operation is problematic; the vast majority of interactive software cannot be effectively verified this way except in the most rudimentary fashion. Additionally, legacy software for ubiquitous Microsoft operating systems is designed to run on a diverse range of hardware conforming to a compatible standard, to which a mature emulation platform such as QEmu or various VMware products (along with the requisite virtual machines, device drivers, and configuration options) comply.

Virtualization platforms developed within the digital preservation community are intended to be flexible, robust, reliably preserved, and readily executed on future architectures. Projects such as JPC and Dioscuri been developed with an emphasis on “write once, run anywhere” code that

is, compiled Java bytecode that can be run in a JVM irrespective of underlying architecture. These projects remain under development, but lack key administrative tools and APIs. A related approach, UVC-based emulation, has at least one implementation that plans support for dynamic executable content [46].

Rothenberg is widely cited for his early arguments describing the necessity to retain original bitstreams, risks associated with format migration due to both imperfect processes and ‘paradigmatic shifts’ [95], difficulties associated with reliance on standards, and principled approaches to emulation. In later work conducted with RAND-Europe for the Koninklijke Bibliotheek (National Library of the Netherlands), Rothenberg provides additional emphasis on maintaining accessibility, and outlines basic strategies for validating “document behavior” on emulated platforms by direct comparison to access on original hardware. As a longer-term goal, Rothenberg describes a ‘stacked’ emulation approach in which outdated emulation virtual machines are described by emulator specifications to be interpreted by their successors [97].

Researchers have explored the use emulation in a variety of preservation frameworks [42]. Some focus on the use of dedicated emulation environments to provide a high level of authenticity or closely replicate the “look and feel” of the original environment, particularly where historical and cultural factors associated with the software are significant [68]. Emulation is also frequently discussed in the context of preservation planning [103]. Until recently, however, there has been a dearth of published technical material on support for end-user interaction with emulated systems.

Larger research initiatives have examined not only the available virtualization tools, but also the formal requirements for their application in a wide range of environments. These include Preservation Layer Models developed for the IBM Digital Information Archival System (DIAS), and ongoing efforts to provide flexible access to a variety of emulation environments in client-server model such as GRATE (Global Remote Access to Emulation) as described by von Suchodoletz [110] and more

recently elsewhere [92].

Additional evidence of the need for automated emulation environments linked to web-accessible archives comes directly from practicing scientists. This is exemplified by nanoHub, a distributed system enabling scientists to conduct simulations in software tools written for legacy platforms via a VNC client accessible via a website. A software API is provided to generate an appropriate interface matching the original input and output requirements. The extensibility of this approach to more interactive applications may be limited.

Migration and Emulation in Practice

Migration and emulation are increasingly viewed as complementary practices for long-term preservation, and this has been reflected in the architecture of both open and commercial archival software systems along with reports published by researchers, national archives, and libraries [44, 81, 56, 111, 94]. Early reports decrying the inadequacy of migration procedures have proved largely unfounded or have been increasingly negated by advances in technology and policy. Jeff Rothenberg's widely-cited Council on Library and Information Resources report [96] claims that "migration is labor-intensive, time-consuming, expensive, error-prone, and fraught with the danger of losing or corrupting information." The first three points are interrelated, and have largely been mitigated by advances in document analysis and processing tools along with increasingly sophisticated registries and format plans [7, 106, 60, 90, 35]. Archives with conservative mandates circumvent the problem of information loss by retaining the original bitstream and associated metadata, which can then be incorporated into an emulation strategy where necessary. Archives which "normalize" files to modern formats on ingest do in some circumstances discard the original binary object, although this practice is increasingly viewed as high-risk.

Migration on request – or the related practice of automated migration initiated by obsolescence

triggers in format registries – is an efficient, low-risk, and access-friendly preservation practice when the original bistream is maintained [67]. For complex or media-rich documents which cannot be easily migrated, emulation (with the access and interface difficulties it entails) is the most viable strategy. Each has merit for particular formats and access conditions, and given the current state-of-the-art neither approach resolves (or will resolve in the foreseeable future) the multitude of issues in ensuring long-term access.

2.7 Managing Risk

There is growing interest in the development and evaluation of risk management tools and techniques to collect, analyze, and disseminate quantitative information on extant and emerging risks to born-digital data [85, 1, 57]. While significant progress has been made, there are numerous open problems in this area. Imperfect and aging hardware, proprietary digital formats and supporting software, limited planning and formal risk management, and accelerating production of digital materials have highlighted limitations in existing infrastructure and in our understanding of what is required to maintain long-term access to legacy digital objects.

Experimental techniques to identify and control for information loss in document collections remain in their infancy. National and international preservation initiatives have focused largely on the development of preservation-specific frameworks, format registries, metadata production and handling, and reliable storage. While many reference extensive internal automation with respect to tasks such as format migration, few provide quantifiable data to support a reasonable level of scientific analysis and verification [85]. Notably, there is ongoing discussion among current practitioners as to whether there are incorrect assumptions about both the difficulty of and the risks associated with digital preservation embedded in our current approaches [99].

Published information on technical risks associated with digital documents remains relatively sparse. In part this is because problems of risk are endemic, and strategies to handle them are complex or depend on unproven (or unrealized) theoretical frameworks. Specific examples of risk include document formatting, inconsistencies in mathematical definitions used in different formats designed to support numerical data, and loss or corruption of metadata during access or modification.

Low-risk, scalable, and cost-effective document migration strategies based on open source technologies are required to provide support for the next generation of digital archives. Many of the risks associated with document migration – even for common procedures such as conversion of legacy Word documents to archival-quality PDFs – are not well addressed in current practice. Furthermore, recent strategies focusing on conversion to flexible migration targets such as XML may not accurately capture the necessary behavior of complex digital objects that depend on legacy supporting software. Documented and undocumented revisions to format specifications and inconsistencies in implementation can produce significant information loss. As a simple example, many of the statistical functions in Microsoft Excel have been debugged or revised over time. While modern proprietary or open source spreadsheet software with the appropriate compatibility filters may be able to open legacy .xls files, the results may not match those seen by the creator.

The following section examines national and international projects which have published data, toolkits, and guides for risk management in large-scale repositories.

Existing Work

Recent initiatives include publications by the Council on Library and Information Resources on information loss in migration of legacy spreadsheets, work on format durability performed by the European Planets consortium, and risk assessment toolkits and environments such as Drambora

and Shaman [50]. Additionally, while environments and tools such as JHOVE and DROID have been developed to validate range of digital formats, they have significant performance issues – including a lack of persistent analysis (immediate failure on encountering problematic objects) and limited format scope.

Common document analysis metrics often have significant limitations from the standpoint of analyzing risk. Some oversimplify or overgeneralize to the point that they do not adequately distinguish between different levels of risk to document preservation. Others rely on expensive and labor-intensive manual inspection and identification of rendering or conversion failure in a workflow that is itself subject to error. Many of the methods described are inherently unscalable, and analysis of document structure and appearance is frequently limited to characteristics that do not adequately reflect the composition of the underlying binary. Consequently, there is significant room for improvement in the development of risk metrics, both in the analysis of distribution of use of various document features and in the degree to which information encoded by those features can be successfully reconstructed.

Future Directions

As migration risk is often path specific, a critical source of information about features likely to result in information loss is operational logs generated by the various translation tools. Such issues are not restricted to legacy formats; both Microsoft and OpenOffice have documented extensive issues in translating between ‘open’ formats such as OfficeOpenXML and the Open Document Format.

One direction in future research will be to lower the human costs associated with risk mitigation in common archival tasks such as migration by providing interactive tools to facilitate evaluation of rendered documents both in their original environments and post-migration. Previous work by the

author provides support for the efficacy of this approach even when using relatively simple tools such as plug-ins for modern office document suites. Such tools may be combined with statistical analysis of large document collections to identify and prioritize documents likely to be “high risk”.

Another method for addressing risks is to synthesize documents that exercise specific features of both legacy binary and modern formats in order to establish profiles for rapid and accurate identification of problem factors in real-world documents. Isolating these features will allow for testing to determine interaction between risk factors, and assist in building additional test cases for archival document management. be debugging rather than research.

Numerous existing publications provide taxonomic descriptions of digital object properties considered to be significant in a preservation context, or those which must be accounted for during transformation (migration from one format to another). A goal of this research will be to provide a scientific exploration of such properties – identifying features of digital objects that affect specific properties, developing processes to automatically identify and analyze those features, testing documents drawn from archival or online sources for the presence of these features, and automating risk-mitigation.

Media Transfer and Bit-Level Preservation

In this chapter I describe methods by which legacy digital collections such as those held on inherently fragile removable media – including but not limited to CD-ROMs and floppy disks – can be “virtualized” through the creation of one-to-one digital copies that are accessible from a shared source via modern workstations. While these methods are presented as a specific technical blueprint for handling issues faced by archives in the present time, they also address a more general problem: providing a basis by which we can build better mechanisms to deal with accelerating technological change. Already the timescale by which we measure the rise and fall of storage technologies and digital formats is measured, at most, in decades. The ubiquitous removable media formats of the period between 1980 and 2000 – the 5.25” and 3.5” magnetic removable disk – are already historic artifacts, no longer in common production and essentially unsupported by modern hardware. The same fate will certainly befall optical and magnetic storage media considered common in the early 21st century; enabling efficient and accurate transfer to state-of-the-art storage media is a first step in ensuring the feasibility of future access.

In the preservation and access scenario presented here, bit-identical images of the original media are served from a distributed file system to provide convenient access on local or remote workstations. The approach utilizes mature open source libraries to provide high-quality, low-cost image extraction, file format identification, metadata creation, and support web-driven access (discussed in Chapter 4). The filesystem images extracted from these media can be variously used to improve search, allow for remote validation and comparison to images drawn from other sources, and improve fine-grained access control.

3.1 Origin and Longevity

Approximately 5,000 CD-ROMs published by the Government Printing Office have been distributed to libraries around the nation beginning in 1987. From a technical standpoint, the CD-ROM offered many advantages: storage capacity hundreds of times that of typical removable magnetic media, low production and authoring costs, and a projected lifespan an order of magnitude above the floppy disks used to distribute some early materials.

While these publications are important sources of scientific, cultural, and historic data, they are sparsely used and present a management problem because they require increasingly obsolete machines and supporting software. These requirements present technical and physical hurdles to casual use, and the materials are themselves are fragile.

Independent sources estimate the lifetime of legacy magnetic media to be as short as 5-10 years¹. While estimates for CD-ROMs are higher (between 10 to more than 100 years for high-quality media), the inexorable pace of technological change will ultimately result in these media becoming inaccessible in all practical circumstances. As an example, images of 5-1/4" floppy disks extracted

¹<http://www.dpconline.org/advice/media-and-formats-media.html>

for this work required installation of a compatible disk drive in an older desktop computer with a motherboard containing both the appropriate controller and onboard connector; the standard machines purchased by Indiana University at the time of writing no longer contain such a controller. While it may be some decades before this is the case for drives capable of reading optical media, many of the CD-ROM material held in existing collections (assuming they have not already been transferred to alternate storage) will be over 30 or 40 years old - well into a period where physical degradation is likely to be a serious problem.

The techniques presented here have been developed through extensive testing during preparation of an online archival collection from the GPO CD-ROMs and floppy disks held at the Indiana University Libraries, and include sufficient technical documentation to facilitate replication at other sites. The majority of these methods can be readily integrated into existing programs or archival work-flows.

The core approach is straightforward; create bit-identical copies of original media for later export through a file server. In this chapter I identify significant technical pitfalls that can readily be avoided. In this chapter I discuss common errors in creating bitwise copies of original CD-ROM and floppy disk media, and outline the use of open source disk imaging and data recovery tools to eliminate these problems.

A number of non-catastrophic error conditions that nevertheless result in information loss can occur during this process [115]. In the following chapter I describe specific methods to reduce or eliminate errors, including identifying and handling image truncation, processing CD-ROMs and floppies originally designed for legacy platforms, and rescuing damaged data. I document failure rates, error conditions, and other risk factors in ISO 9660 format images (and raw floppy images) extracted from the IU GPO collection using these techniques, and describe the specific advantages

of open source data imaging and recovery tools. In addition, I discuss avoiding common incompatibilities in handling discs mastered for a variety of legacy platforms.

These techniques form a core set of best practices in the transfer of data from legacy media, and provide a flexible way to handle a wide range of access scenarios discussed in the next chapter.

3.2 Imaging Removable Media

A critical assumption of handling these materials is that a bit-faithful image of the optical or magnetic media preserves all of the useful properties of the original media while enabling distributed and web-based access to the materials. In contrast, the original physical media are subject to degradation and limit patron access. This premise is valid except for materials with copy protection schemes that violate the underlying data storage standard or depend on physical characteristics of the media.

Extracting bit-faithful images from optical and magnetic media is a relatively common task, and many free and commercial software applications exist to create such images for alternate storage. In large collections, many of these programs can introduce “silent errors” or fail for non-compliant or even moderately damaged media. I examine how to perform operations of this type in a manner that minimizes the risk of introducing error and corrects for errors that may have been introduced during creation of the media. This approach is supported by experiments performed on over 4000 removable media items in the IU Libraries collection. I focus primarily on optical media; however, the same issues exist with magnetic media and the same tools used to image optical media may be used to image magnetic media. I provide additional data on success rates extracting data from legacy floppy disks in the GPO collection.

Removable optical and magnetic media have a limited shelf-life. Degradation of the media can

occur due to incorrect or inadequate storage, damage during handling, and wear on the media during access. Proper care and handling of the media itself is discussed extensively in the related literature [47]. ‘Media refresh’ or disk-cloning is a labor- and materials-intensive process that can produce cumulative error.

Retaining information on legacy physical media is an impediment to access, typically limited to a single user on a local workstation at any given time. Conversely, bit-faithful image files can be accessed simultaneously by multiple independent users and can be presented to patrons in a file system view which is abstracted away from the actual storage device used to retain the objects. Image files can be natively mounted as virtual hardware devices in Linux, OS X, most emulation platforms, and under Windows with readily available software.

A key question is what constitutes a bit-faithful image. CD-ROMs in particular consist of raw data organized as a sequence of fixed-sized sectors. Overlaid upon these sectors and specified by the ISO 9660 standard is a file system.² Due to a variety of factors in the mastering and writing processes, a CD-ROM may include extra sectors that are not part of the file system. Our experience suggests that exactly the set of sectors referenced by the ISO 9660 file system need be preserved – indeed only these sectors are accessible by normal application software. There are many practical issues involved in capturing the relevant sectors in a disk image.

Major risk factors in the creation of ISO images include lack of conformance to the ISO 9660 standard in the original media, and errors introduced by the ISO creation software – either due to malformed data or bugs in the software. These errors can be difficult to isolate; in the following section, I outline some common problems observed in imaging the GPO collection. Additionally, I examined the performance of a variety of proprietary and open source ISO extraction software packages on Linux and Windows in order to provide some guidelines for successful ISO image

²<http://www.ecma-international.org/publications/standards/Ecma-119.htm>

creation.

Verification and Common Errors

Basic information on the contents, metadata, and integrity of ISO filesystems can be extracted using the `isovfy` and `isoinfo` tools, although they do not identify all possible technical issues that arise with these images. For a well-formed image, the `isovfy` output will appear as follows:

```
[1655][svp@troy] isovfy 30000094726506.iso
Root at extent 14, 2048 bytes
[0 0]
No errors found
```

The `isoinfo` tool can provide more detailed metadata about the volume, including information from the primary volume descriptor, extensions present, or a complete listing of the path table. The volume descriptor for the image verified above appears as follows:

```
[1655][svp@troy] isoinfo -d -i 30000094726506.iso
CD-ROM is in ISO 9660 format
System id: CDEVERYWHERE
Volume id: HUS03APR04
Volume set id:
Publisher id: US GPO
Data preparer id: CLAUDINE BYNUM
Application id:
Copyright File id:
Abstract File id:
Bibliographic File id:
Volume set size is: 1
Volume set sequence number is: 1
Logical block size is: 2048
Volume size is: 129175
Joliet with UCS level 1 found
Rock Ridge signatures version 1 found
```

Some of the common issues encountered with ISO 9660 formatted CD-ROMs are described below. This list reflects issues encountered in preparing the GPO collection; it serves to illustrate

some of the primary risks that may not be well handled by some commercial software, or require additional processing of the filesystem for later verification and comparison to secondary copies.

Physical damage. Physical damage cannot consistently be evaluated with a visual inspection. In testing the GPO collection, only one CD-ROM out of thousands imaged had obvious physical damage (a radial crack) that precluded imaging. The most prevalent form of physical damage was surface scratching incurred during storage, typically where plastic cases had become damaged and no longer supported the disk in a stationary location. Because the task here was not physical management or preservation of the disks themselves, such damage was ignored except where it was severe enough to interfere with imaging. In such cases, disk imaging was reattempted using the `ddrescue` program.

File truncation. Following extraction, some ISO images may exhibit problems with access to specific files within the path hierarchy, in spite of the fact that the image can otherwise be mounted and appears well-formed. Typically, this will manifest as an error when attempting to open a file that appears in a specific directory. File truncation may be due to incorrect authoring of the image, damage to the physical media, or incorrect extraction of the filesystem from the disk. These errors can be traced programmatically through the use of libraries available as part of the UNIX `cdio` package. Beginning at the top of the path tree, a small program can be constructed to “walk” all entries at each directory level (including the directory entries themselves), comparing size-on-disk to the recorded size for each entry in the filesystem. An example of such a program is provided in Appendix A.

Incorrect block size. The CD-ROM Mode 1 specification (used for computer data, and appearing most frequently) is comprised of entities known as “sectors” and “frames”. Each sector holds 98 frames, and each frame holds 2352 bytes. Of these bytes, 2048 are set aside for user data. The remaining bytes are used for synchronization, header data, and error correction. In the ISO 9660

filesystem standard, block size is 2048 bytes (aligning with the user data for each frame). Some CD-ROM authoring software (particularly some early commercial authoring programs) allow this block size to be respecified without warning the user; this can cause compatibility issues with certain legacy software and hardware.

Bad length in volume header. In most modern CD-ROM authoring software, the volume length header is recorded to reflect the actual length of the volume reflected by the completed write operations. However, in legacy collections experience suggests that a small but significant percentage of the ISO filesystems will be recorded with an incorrect volume length. There may be several reasons for this. Some legacy authoring programs simply did not record the correct volume length at any time, resulting in a default value that is typically less than several hundred blocks. Most modern operating systems (and all UNIX/Linux variants we tested) are not affected by this, as they follow the actual path structure of the filesystems irrespective of the volume metadata. However, some early distributions of Windows and even some current Windows-based authoring and image-extraction software will display an error when encountering these values.

TAO (1 sector short). The ISO 9660 standard allows for multiple volumes to be recorded to CD-ROMs on different tracks. When a single volume is recorded in “track at once” mode, CD-ROM authoring packages will “mark” an extra (empty) block, in order to indicate the end of the track. While this does not generally cause access or compatibility issues, some image extraction software offers the choice of correcting the volume length advertised in the header; if this action is performed, it should be recorded in associated archival metadata.

3.3 Filesystem Processing

A fundamental problem is that without more than one copy of a given CD-ROM, it is impossible to guarantee that a disk image contains the correct bits. At the sector level, CD-ROM data is protected by a 3-layer error correction and detection scheme (2352-byte sectors containing 2048 bytes of data, 4 bytes of error detection, and 276 bytes of error correction). This is interpreted by the CD-ROM hardware to compensate for damage or degradation. While the probability of an undetected bit error is small, it is not impossible. However, this problem exists whether one utilizes the original media or an image of the media, and – in the former case – is likely to increase as the materials degrade.

CD-ROMs are accessed by the operating system as “block devices”, in which sectors correspond to fixed-sized blocks in a single binary file organized as an ISO 9660 filesystem. The ISO filesystem is constructed of one or more volumes beginning with a dedicated sector, called a volume descriptor. The volume descriptor includes an identifier, volume size in sectors, sector size, and pointers to directory information within the volume. The directory information includes *path tables* (a rarely used mechanism for quickly finding files) and a *root directory*. As with most file systems, directories are implemented as binary data structures embedded in ordinary files.

The simplest method to determine ISO image length is to examine advertised volume length for each volume in the filesystem. This is risky, however, as volume header information is frequently wrong – 19% of the CD-ROM images created had incorrect size information (too long or too short) in their headers. In “track at once” recordings, images are typically one sector shorter than advertised. The advertised length can also be too high when the recording software computes the image size prior to “compacting” the file system.

This problem can be overcome by walking the file system directly, computing the starting sector

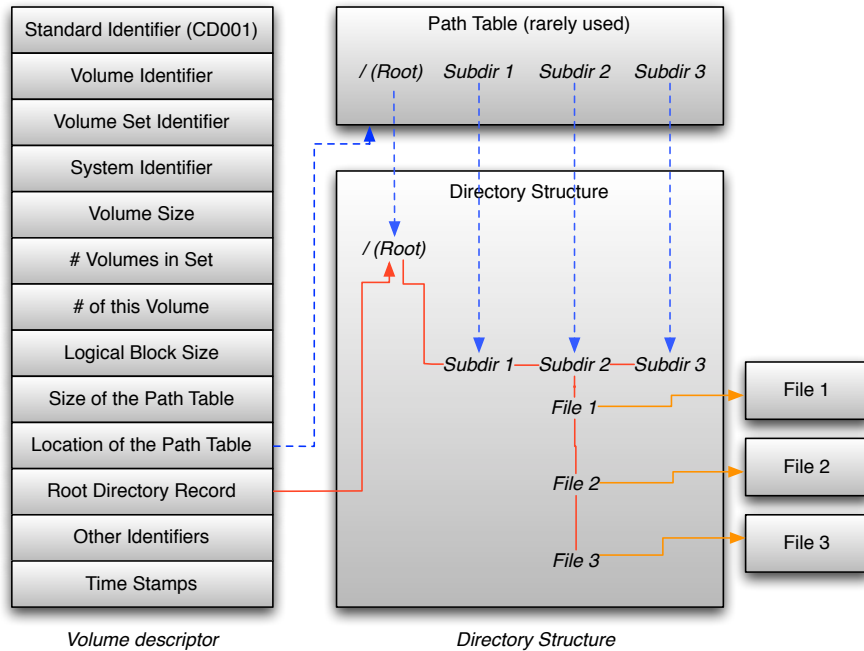


Figure 3.1: ISO 9660 filesystem organization

and length of each file individually. By performing this type of filesystem walk with a simple C program using the `libiso9660` library, it was apparent that approximately 8% of the images created with Windows based software were truncated before the end of the image. Experiments with a variety of windows based tools on multiple machines confirmed this behavior. In contrast, the Linux tool `dd` enables copying all of the raw data from a CD-ROM. In a typical case, `dd` would be executed without modifying arguments on the command line beyond specification of block size and destination:

```
dd if=/dev/source_cdrom of=/data/extracted_image.iso bs=2048 conv=sync
```

In an experiment with 86 CD-ROMs whose images were truncated by Windows, complete ISO images for 81 of these could be extracted using `dd`. Of the remainder, 4 exhibited minor damage

resulting in lost sectors, and 1 was physically damaged to an extent that prevented any recovery. The rate of failure to read all the bits of CD-ROMs should remain 1% provided the correct tools are used.

A consequence of the variability in behavior of ISO extraction utilities is that simple MD5 checksums on images extracted from two CD-ROMs may not match even when the user data contained in the filesystem is identical. A direct walk of the filesystem can aid in reliably determining whether two images do in fact correspond to the same data source.

Failure rates on floppy images are expected to be somewhat higher due to sensitivity of the magnetic media and the lack of explicit error correction. The majority of legacy floppy disks have a relatively simple (FAT12) filesystem organized in clusters of 1 or 2 512-byte sectors. We used `dd` to extract `.img` files from 5.25 and 3.5 inch floppies, and observed failures in 21 out of 381 total images. The majority of these were from older low-density 5.25 disks, for which extraction was attempted using new hardware purchased to limit the introduction of error from worn drive mechanisms. An extensive survey of floppy disk extraction was previously performed for the CIC Floppy Disk Project, now available via the GPO website [28]. As these materials continue to age, specialized hardware or forensics software may be required to minimize error rates and damage to the media during read cycles.

In some cases, read failures due to physical damage can be overcome using data recovery tools such as `ddrescue`, which also provide the ability to reconstruct a 'clean' copy of a data item from multiple damaged media sources. In typical cases, `ddrescue` can be run first to extract those data sequences of the image that are undamaged, and in a following read perform an attempt a block-level extraction of damaged sections. An example based on the GNU `ddrescue` manual:

```
ddrescue -n /dev/damaged_original /dev/partial_extraction rescued.log  
ddrescue -r 1 /dev/damaged_original /dev/partial_extraction rescued.log
```

With these cases, we can build a risk-averse strategy for the identification, imaging, and if required reconstruction of CD-ROM images:

1. *Extract* ISO 9660 image using low-risk utility (`dd`)

If sector errors are reported, re-extract (`dd / ddrescue`)

2. *Analyze* image

Walk the filesystem. Is there truncation?

Check the volume length. Does it match the raw sector count?

3. *Update* image

If truncated, attempt image rebuild with `ddrescue` and secondary source (if available).

Rewrite header info to correct volume length if required.

Examination of volume-specific metadata uncovered a high degree of variability in authoring procedures used to create CD-ROM materials for the GPO collection. Our tests indicated several cases where early commercial premastering software such as OMI QuickTopix produced incorrect volume lengths even when an image was otherwise correctly mastered. Such mismatches can confuse ISO extraction packages that assume this metadata is always correct.

We conducted additional ISO extraction trials using the proprietary software packages IsoExtractor v2, MagicISO, PowerISO, Image Master, Undisker, Alcohol 120%, and the Forensic Acquisition Utilities on a Windows platform. We also examined a Windows clone of the UNIX `dd` package³ and standard GNU `dd`.

³<http://www.chrysocome.net/dd>

Software	Warns of bad volume length	Copies valid ISO when vol. length bad	Prompts to update metadata
IsoExtractor v2	No	No	No
MagicISO	No	No	No
PowerISO	No	No	No
Image Master	No	Yes	No
Windows DD	No	No	No
Forensic AU	No	No	No
Undisker	Yes	Yes	Yes
GNU dd (Linux)	No	Yes	No

Figure 3.2: Common imaging utilities

The majority of proprietary software packages exhibit high-risk behaviors. Of the proprietary software examined, only Undisker was able to identify and present to the user options for transformative and non-transformative copies of CD-ROMs encoded with incorrect volume length data in the header. This is somewhat remarkable, given that Microsoft provides clear developer guidelines for low-level disk access under Windows.⁴ Note that while `dd` does not explicitly warn of bad volume length, it does not depend on the header information, but rather copies raw sectors until the end of the image is reached.

These trials provide an indication of the immediate risks in relying on proprietary or even unverified open source imaging software. Even when sophisticated checksumming and data-loss prevention controls are in place, a simple inability to read the correct size of the media can result in catastrophic error. Such errors can and should be prevented by ensuring that a given tool is capable of performing both a low-level analysis of the media and a complete walk of the filesystem.

⁴<http://support.microsoft.com/kb/138434>

4

Collection Processing, Metadata, and Access

The process of analyzing and transferring bitstreams from legacy media to modern storage systems – along with conversion of existing metadata to preservation-friendly modular metadata technologies – is generally viewed as an expensive, error-prone, and labor-intensive task [17]. One of the goals of this work has been to demonstrate that careful planning and automation can lower the associated costs, reduce instance of error, and improve facilities for identifying and correcting errors that appear in the resulting collection.

Media transfer of the GPO collection analyzed in this work was largely performed over the course of three years by a single part-time work-study student. This student (AF) assisted in building a collection of 4233 items corresponding to 4199 CD-ROM images and 360 floppy images. Her total labor was 832 hours, corresponding to 12.5 minutes/item (roughly 7.5 minutes/image) including the rework required due to process failures and faulty images. This chapter begins with an elaboration of the process and identifies specific insights gained.

The key interface between the research team and AF consisted of a networked file system, where AF deposited all created images, and a web accessible database, where she recorded key information about the created images. All work was tracked at the item level using the Codabar label

number [18]. Files created from a single item were named using the Codabar number, a suffix indicating the file number, and a suffix for the file type (e.g. `number.1.iso`, `number.3.img`). Each item was represented by a single database row including the Codabar number, the corresponding catalog number and title, a library catalog “key”, and notes created by AF during the copying process. Each key was then used to access the MARC record of each item through the Indiana University library z39.50 server. Z39.50 is a standard maintained by the Library of Congress, typically used as part of an integrated library search and retrieval system to locate bibliographic data. While the protocol used for search is sophisticated (supporting complex queries based on six attributes that a client can use when making requests to the primary server), it is unwieldy in modern Web-based environments. The SRU/SRW protocols (respectively, REST and SOAP based) have been developed to address these problems, maintaining the query syntax but replacing the original protocol with HTTP [63].

The database was initially populated with a spreadsheet of GPO publications provided by library staff. Subsequently, AF added items to the database manually. Ideally, this access to library catalog information would be automated, but the research team did not have direct access to internal library systems. The database was implemented with MySQL and a web interface created with phpMyEdit. The database interface automatically populated a date field to enable work tracking.

AF worked largely independently. At each session, she used the database to determine, in catalog order, the next group of items to copy, pulled these items from the stacks, and copied the corresponding media. For each item processed, she entered notes in the database as well as the number of image files created. As image files were created, she transferred the files to the research file server using Windows file sharing. AF routinely checked the database for missed items, and augmented the database with newly cataloged items (helpfully segregated by the library staff).

The research group periodically checked the created image files for errors and copied valid

images to a distributed AFS filesystem implemented with OpenAFS [82]. AF's work was tracked through a script which generated reports and sent weekly emails to AF and the research team. These reports included statistics about items copied per month as well as a list of items needing rework due to missing or incorrect files. While the work performed by AF was relatively labor intensive – requiring acquisition of CD-ROMs from library holdings, transfer to the imaging facility, and recording of the work performed – the process is both efficient and replicable at other sites. In a pilot project conducted by researchers at New York University and Yale University, the analysis and migration cost for an individual CD-ROM was calculated as \$19.25 at a student pay rate of \$11 per hour (2007 USD). Notably, this rate included student generation of an XML conversion of existing MARC metadata – a process that is handled automatically in the scheme presented in this research. At 7.5 minutes to completion for each CD-ROM in the GPO collection held at the Indiana University Libraries, the cost of media transfer is likely less than a quarter of what was calculated for the pilot.

The collection materials were organized in volumes on the AFS file system using a directory for each item; the directory was named using the Codabar number and had permissions assigned consistent with usage restrictions (while all GPO items have public access, other materials are restricted). Each directory consists of a set of image files and could, in principle, include metadata or other files. For example, it would make sense to scan all the printed materials associated with an item to a single PDF file.

During this process, it became clear that specific care was required not only in creating the bit-faithful images, but additionally in routinely checking the images for quality and a requirement for automated reporting structures. A large number of bad images were identified – approximately 8% – using the initial set of copying tools. We subsequently developed techniques to detect bad images and AF had to find the corresponding disks to perform rework. These are described in detail in the

previous chapter.

Automated reporting scripts were used to track progress while constructing the collection. Initially, the used the ISO verification tool `isovfy` was used to identify errors. However, in many cases it provided incomplete or incorrect information. For example, many of the images were reported as containing errors related to the use of Rock Ridge extensions (which provide support for Unix-specific file information on ISO 9660 CD-ROMs) even when no Rock Ridge encoding was present. Certain images were identified as having directory structures with ‘unusual’ sizes. By walking the ISO filesystems directly using the `libiso9660`-based tool described previously, we were able to provide AF with regular feedback on issues with the processed items and the research team could monitor progress.

4.1 Testing the Collection

A part-time work-study student (KM) tested ISO images in order to profile usability and compatibility on a workstation configured to reflect the GPO “Minimum Technical Requirements” for local access workstations at holding institutions.¹ This consisted of a Windows XP Professional virtual machine with Microsoft Office 2003 and Adobe Acrobat installed. The machine was configured to access ISO images on a virtual D: drive, and restored to an unmodified state after each test.

KM examined the contents of each ISO and recorded notes on a number of accessibility factors, including the success or failure of required installation procedures, whether software required to view specific file formats was included with the original CD-ROM, and any other failures related to contextual dependencies or apparent damage to the image. An overview of this data is provided

¹<http://www.fdlp.gov/administration/computers/244-mtr>

in Figure 4.1. Note the numeric overlap in the application tests. For example, access to information on a given ISO may require both a browser and additional binaries.

Factor	Items
No install required	734
Successful install	581
Successful modified install	56
Install fails	11
Requires Web-browser	153
Requires MS Office	331
Requires Acrobat	405
Requires other commercial	94
Custom binary provided	930
Good documentation	1119
Minimal/No documentation	263
<i>Total items tested, each category:</i>	1382

Figure 4.1: Testing GPO CD-ROM access.

At the time when aspects of this work were originally published, KM had reviewed 1382 items from the collection corresponding to more than 1600 individual ISO images. 11 images exhibited unrecoverable errors preventing installation or execution of included software. 3 of the 11 failed due to memory access issues in Windows' DOS-compatibility mode. 2 images were identified as corrupt. The remaining 6 contained software installations that could not be completed in the test environment.

56 images contained installation procedures requiring manual modification. For example, update of an initialization file containing local variables specifying the location of the drive from which to install. Discounting common office formats (Word, Excel, PowerPoint) and document viewers (Adobe Acrobat), 94 additional images required software packages not distributed with the CD-ROM. These included commercial binaries such as ArcView, and custom OLE controls.

4.2 Accessing the Collection

The FDLP collections were originally organized as file systems with independent directory structures. A collection was used by mounting its file system (using removable media) on a suitably configured workstation [77]. A user could then explore the file system, open documents and data with appropriate applications (often included on the media), and execute programs provided on the collection media. This file system structure is important in preservation because many of the files (e.g. HTML) depend upon the original path structure to be correctly rendered. Consequently, our current system is organized around a *virtual file system*. As described in Section 5.2, a user browsing a collection is presented with a view which may include both the original collection files as well as migrated renditions of these files and contemporary usage information and metadata. These dynamic views are generated from bit-faithful images of the original collection media. A significant feature of our system is the capability for both privileged and non-privileged users to contribute to the existing metadata and usage information presented by our system.

We expect most accesses to individual collections will be aimed at determining suitability to a particular researcher's goals. This task will be simplified by the ability to "peek" within files regardless of their original format. In cases where access to the original files or migrated renditions is insufficient due to limitations of migration tools or lack of access to contemporaneous software, virtualization provides another level of security. Finally, access to the original media ensures that the preservation mechanisms provided do not themselves limit future use.

Architecture

The system consists of two major logical components: a *virtual file system* which maintains all of the data and metadata for the various collections, and a *web application* which enables access to

this file system. The virtual file system is organized at the collection level with subdirectories for media images, links to mounted images, static files such as usage notes and metadata, and dynamic files such as those produced by migration tools. The system also supports browsing within archive formats such as TAR and ZIP.

A custom web application provides user control over the file system view, and provides privileged access to modify supplementary files such as metadata. The basis for preservation in our work is bit-accurate ISO (ISO-9660) images of CD-ROMs and DVDs – all renditions are derived dynamically from these images. Support for browsing ISO images is a standard part of Linux and BSD. In early implementation tests, we used auto-mounting to manage mounting images on demand; however, because triggering kernel events is undesirable from a security standpoint, this was later replaced with a C program based on the `libison9660` library to extract filesystem contents directly from the ISO image.

A session view is created by stacking the various file system directories to form a single *union* which merges files within matching directory paths with permissions derived from session information. These stacked directories include *synthetic* files that are generated on access including migrated renditions of legacy files. The idea of utilizing synthetic file systems to present dynamic data has a long history in operating systems, most notably in Plan 9. [53, 87]. Union file systems have been widely implemented and are used for applications such as “live CD” Linux distributions that enable execution of an operating system without installation [86, 116].

As an example of this file system model, consider Figure 4.2. The “root” directory of a collection contains four subdirectories: “media” containing raw ISO-9660 images, “files” containing mounted versions of each of the raw images, “static” containing catalog metadata (marc.xml), usage notes and additional files associated with the mounted images, and a “dynamic” directory with files

derived from the original collection data.²

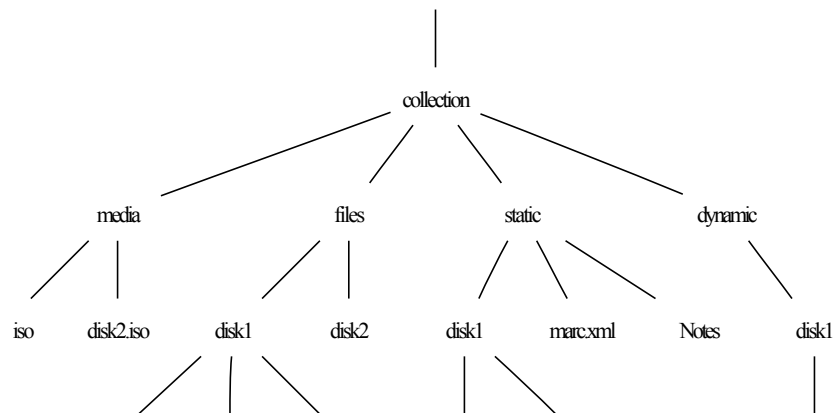


Figure 4.2: Example “File System”

In the GPO collections there are many obsolete files and directories. Examples include applications such as Internet Explorer for which there are more modern versions available, or installation directions that are outdated with respect to more recent operating systems. In keeping with our principle of deriving views from original media images, we elide these directories through the use of *whiteout* files. These whiteout files may be created both statically by system administrators and dynamically based upon the current session. For example, it may be necessary to block access to certain files for copyright reasons; this requirement can be implemented for all users with a static whiteout or dynamically based upon current session attributes including preferences, location, and user identification. Returning to our example, a session with no access restrictions and with migration enabled might be logically organized as illustrated in Figure 4.3. The actual presentation of this view is provided by the web application which generates root pages for each collection from the cataloging information and contributed metadata.

²For performance reasons, it may be desirable to pre-compute and cache some dynamic files

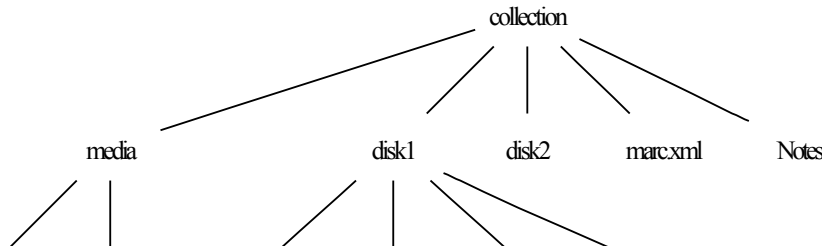


Figure 4.3: Example “File System View”

A key design decision that emerged from preliminary tests was that – given a URL – it should be possible to “walk up the tree” to determine context. For example, consider the URLs illustrated in Figure 4.4. All collection specific information is stored in METS records. We format parts of these records as HTML for catalog records, determine the locations of individual media images from the METS records, and index these records to enable searching. In this example, 4909070 is the index of a specific METS record, FID1 is the index of an (ISO) image file referenced by that METS record, and FID1/high_res/cgro.mov is a file within that image.

Server Root	<code>www.cs.indiana.edu/svp/</code>
Search Page	<code>http://www.cs.indiana.edu/svp/search</code>
Formatted METS record	<code>http://www.cs.indiana.edu/svp/4909070</code>
Root of CD-ROM image	<code>http://www.cs.indiana.edu/svp/4909070/FID1/</code>
File in CD-ROM image	<code>http://www.cs.indiana.edu/svp/4909070/FID1/ low_res/cgro.mov</code>

Figure 4.4: Namespace

Implementations

Two implementations were created to reflect these design criteria. One version of the system was created using the Django web development framework [22]. This incorporated a variety of experimental features including user-submitted metadata modifications, dynamic whiteouts (for

example, removing irrelevant or undesired filetypes from the users view), IP-based content restrictions, and fully on-demand migration services based on user selection. Additional information on this implementation is provided in Chapter 5. While use of the Django framework allowed for the introduction of many desirable features, the codebase imposed a considerable overhead on maintenance and portability. As one of the goals here was to produce lightweight, scalable tools that could be used in a variety of contexts, a separate implementation was created for the live site.

The final site is organized as a Perl CGI script that utilizes custom binaries to extract files and directories from images, and an off-the-shelf tool (SWISH-e) to index and search the collection. The information used by the CGI script to generate pages includes the SWISH-e index, the METS XML records, and XSLT style sheets to extract and format information from these records. The site code consists of 660 lines of Perl, 182 lines of XSLT, 300 lines of CSS, 161 lines of HTML, and 152 lines of template code (used by the Perl to generate pages). Tools to support the site include scripts to generate METS from MARCXML, to pull MARCXML from the Indiana University Library x39.50 server, and to index the METS records.

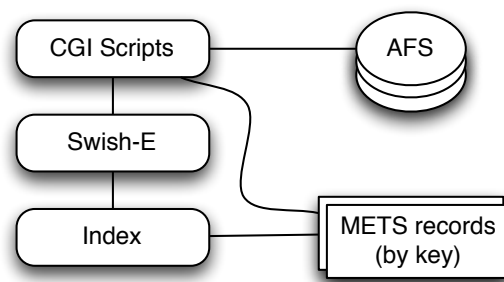


Figure 4.5: Metadata handling

Metadata Creation

Web access to the collection is managed through METS records containing key bibliographic information derived from the library MARC records and item specific information including catalog numbers (e.g. SUDOC numbers for GPO items) and files locations. Both the bibliographic information and item specific information are in MODS form. Consider the METS record in Figure 4.6 corresponding to the Patent office gazette. This record includes bibliographic information for a series of items, item specific information with an ID derived from the Codabar number, and links to the files corresponding to the item.³

The web service processes METS records. SWISH-e serves as a high-performance indexing tool to process these METS records and all displayed bibliographic information is dynamically generated from relevant fields within the METS records. The collection is updated by copying new or modified METS records to the web server and regenerating the corresponding index.

The automated process for generating METS records for the collection involves the following steps for each unique catalog key in the database.

1. Pull the corresponding MARCXML record from the Indiana University z39.50 server
2. Translate MARCXML to MODS with a “crosswalk” stylesheet.
3. Generate common part of METS record
4. For each corresponding item generate a bibliographic entry
5. For each corresponding item generate file information

Access to the z39.50 server was automated through a small C program built with Index Data’s YAZ toolkit [117]. Each database entry included an internal IU catalog key used to generate a z39.50

³The entire METS record can be accessed at <http://www.cs.indiana.edu/svp/mets/5705037> and the generated web page at <http://www.cs.indiana.edu/svp/5705037>.


```

-- (Bibliographic Information)
<mods:mods>
  <mods:titleInfo>
    <mods:title>Official gazette of the United States Patent
      and Trademark Office</mods:title>
    <mods:partName>Patents</mods:partName>
  </mods:titleInfo>
  <mods:titleInfo type="alternative" displayLabel="Title on
    welcome screen:">
    <mods:title>Electronic official gazette for
      patents</mods:title>
  </mods:titleInfo>
  ...
-- (Item Specific Information)
<mods:relatedItem ID="ID30000053708008">
  <mods:classification authority="">C 21.5/6:
    v.1263,no.3 2002</mods:classification>
  <mods:note type="system details">
    ...
</mods:relatedItem>
...
-- (Files associated with an Item)
<mets:fileGrp USE="CDROM Image">
  <mets:file ID="FID7" MIMETYPE="application/x-iso9660"
    DMDID="ID30000053708008">
    <mets:FLocat xlink:href="file:///afs/iu.edu/
      public/sudoc/volumes/05/30000053708008/...
  </mets:file>
</mets:fileGrp>

```

Figure 4.6: Example METS Record

query. The retrieved MARCXML records were converted to MODS with a lightly modified version of the Library of Congress MARCXML to MODS stylesheet.⁴ These records were then processed using Perl scripts to build METS documents containing links to the media images and to build a searchable index.

Metadata presentation and collection access is handled only at the level of a full ISO image within which relevant search results are contained. In an early prototype of the system, a facility

⁴<http://www.loc.gov/standards/mods/v3/MARC21slim2MODS3-3.xsl>

was included to allow additional usage metadata to be collected directly from the Web interface via a small AJAX script. It was envisioned that users accessing the collection could add comments and operational or technical suggestions for specific items within an image.

Navigating within a given ISO image, the user interacts with a hierarchical file-folder interface similar to the default presentation of static directory trees by Apache. However, this view is moderated according to the existence of “white out” information, user access permissions, and availability of migrated files. For example, a directory containing an outdated version of Adobe Acrobat may not be displayed to the user. A series of files with location-based restrictions likewise may not be displayed to a user on an inappropriate subnet, or within a restricted IP range. Those files for which migrated renditions are available are displayed with an additional icon linked to the migrated file. In the Django-based implementation (described in further detail in Chapter 5), a simple AJAX routine is used to poll the readiness of migrated items for those which are handled on-demand.

Server-side modules and utilities for file migration may operate independently of the web-based interface as daemons in full-batch mode or be triggered by user navigation. In the most recent implementation, migrated files are saved following a batch operation to reduce overhead on the relatively limited hardware used. In the first implementation, a model within the Django framework specifies those file types that are candidates for migration and whether the migration for a type should occur synchronously - for each file on request from the server - or asynchronously. Asynchronous migration requests are triggered the first time a user navigates to the top-level directory for a particular ISO image. All files with types corresponding to asynchronous migration requests within that image are migrated while the user continues to navigate the archive (or alternatively, pre-computed through batch processing).

Access Within ISO Images

Numerous access issues arise in handling legacy CD-ROM images. Files are frequently in obsolete formats. Many files contain links that implicitly depend upon the ISO image being mounted as a virtual CD. ISO images created on legacy platforms may not be fully supported on the host system. Standard ISO 9660 extensions such as Joliet and Rock Ridge (intended to overcome file naming and metadata issues in the original ISO 9660 specification) can make it difficult to correctly render all file names and utilize these rendered names to find files in ISO images.

Some CD-ROMs created for Macintosh computers have compatibility issues. This issue is manifested in pairs of identically named files representing *resource* and *data* forks intended to separate structured and unstructured data. While the end user is generally interested in the data fork, Linux is unable to extract the correct file from an ISO images – consequently, it was necessary to patch `libiso9660` to “do the right thing.”

Our web site allows patrons to browse within an ISO image as a virtual tree within the server file system. This browsing capability is supported by two small custom binaries that extract files and directories from ISO and IMG (floppy images) files – `floppyextract` for floppy IMG files and `isoextract` for CD-ROM ISO files. These programs were created using the `libiso9660` library for ISO images and the `libsharedmime` libraries to support determining file type; they required approximately 1200 lines of C and C++. Each of these binaries takes two arguments – the location of an image file and a path within that image. They return either a csv directory structure or a raw file depending upon the object referenced by the path. In the case of a raw file, they also return a mime-type suitable for responding to an http request. For example, the image referred to as FID1 in Figure 4.4 is located through the METS record as `/afs/.../30000078894163.iso` (path elided for space). The command:

```
isoextract /afs/ ... /30000078894163.iso/low_res
```

Returns the following record:

```
DIR  
file,video/quicktime,cgro_low.mov,957892418, 3932717,cgro_low.mov
```

The first line indicates whether this is a directory (DIR) or file (FILE). In this case, the directory contains a single file, of type video/quicktime, named `cgro_low.mov`. The two integers are the encoded permissions and creation time. The repeated name is provided to simplify the CD-ROM lookup – in many cases, the displayed name and internal name differ due to issues relating to Joliet and Rock Ridge extensions to ISO9660. Our web server is driven by a CGI script that uses the output of these programs to provide a user view of the file systems embedded in CD-ROM and floppy image files.

Reliable file format identification is important in this environment, as we wish to provide the user with a consistent view of the filesystem, in addition to support for format migration [114]. As noted previously, the `libsharedmime` library is used to report file types based on matches in the Shared MIME-Info database. The `shared-mime-info` package which provides the core database can be easily updated to accommodate specific collections or missing file types. Output translations exist for more than a dozen languages. Standard `libsharedmime` implementations provide built-in support to return MIME-lookups, and can fall back or verify based on glob (regular expression match) patterns on filenames or execute a magic-number lookup.⁵ These implementations are fast, platform-agnostic, and provide the breadth of coverage required to deal with large, heterogeneous collections of files.

⁵<http://www.memecode.com/libsharedmime.php>

4.3 Access Overview

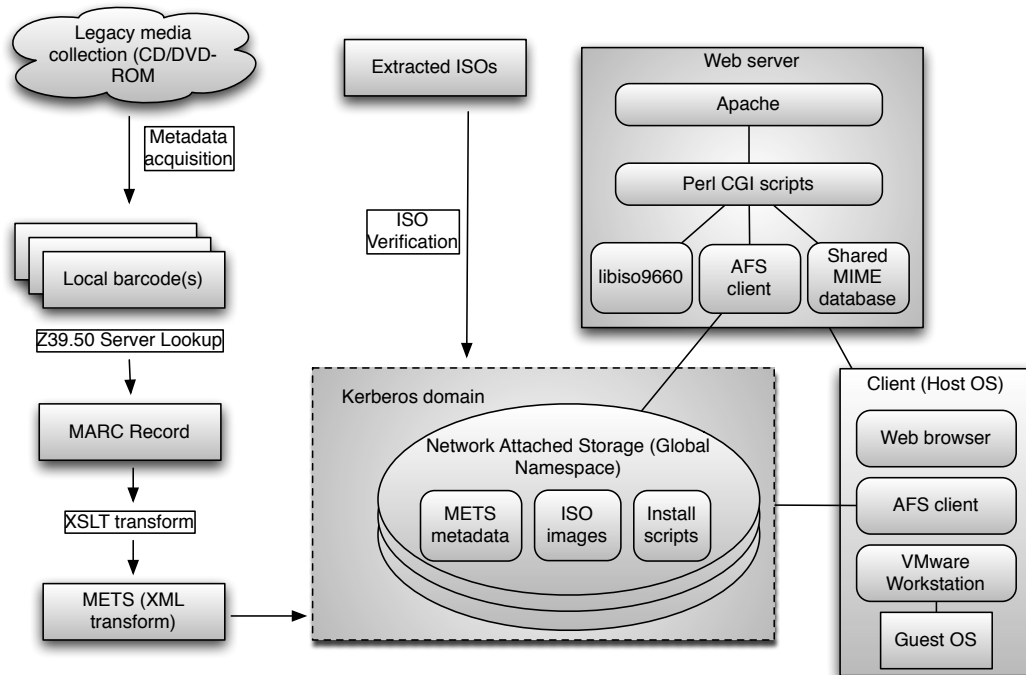


Figure 4.7: System overview

An overview of image acquisition, storage, and access can be seen in Figure 4.7. Note that this presents the a high-level view of both the preprocessing and access mechanisms (for example, no path is shown for the index, which can be automatically rebuilt as required), and reflects the organization of the second (CGI-based) implementation described earlier. Most of the modules shown here are transparent to the end user, with the exception of the AFS and virtualization clients which must be installed in order to facilitate use of an emulated environment without the necessity of downloading a complete ISO image.

A user navigating to the SVP site at <http://www.cs.indiana.edu/svp/> is presented with a search interface through which they may enter simple search terms or more specific classification

identifiers such as SUDOC numbers. On executing a search, the user is presented with a list of relevant items. Selection of an item takes the user to a page whose corresponding URL contains the item number and the content of which consists of a human-readable transcription of the bibliographic metadata, links to download or browse the ISO, and a link to the full raw METS metadata record, as seen in Figure 4.8. Additional links are provided to the original catalog record held by the Indiana University Libraries and any installation notes created during the collection survey.

Home

Search

List All

Migration Examples

Climatic atlas of the Barents Sea, 1998 : temperature, salinity, oxygen / Murmansk Marine Biological Institute (Russia), Ocean Climate Laboratory, National Oceanographic Data Center (USA). Murmanskiaei morskoai biologicheskiaei institut ; National Oceanographic Data Center (U.S.) Ocean Climate Laboratory. Washington, D.C : U.S. Dept. of Commerce, National Oceanographic Data Center[1998]

Notes : *NODC-121*--Disc label.
Text in English and Russian.

Subject Headings : Oceanography -- Barents Sea -- Charts, diagrams, etc
Deep-sea temperature -- Barents Sea -- Maps
Water -- Dissolved oxygen -- Barents Sea -- Maps
Salinity -- Barents Sea -- Maps

Record Info : (OCoLC)ocm42011634
(GPO)99056140
(InU)AL55301FW
(InU)BNK5342NW
(InU)ALP212788

[Indiana University Catalog Information](#)

C 55.297: C 61/V.1/CD

[Installation Notes :](#)

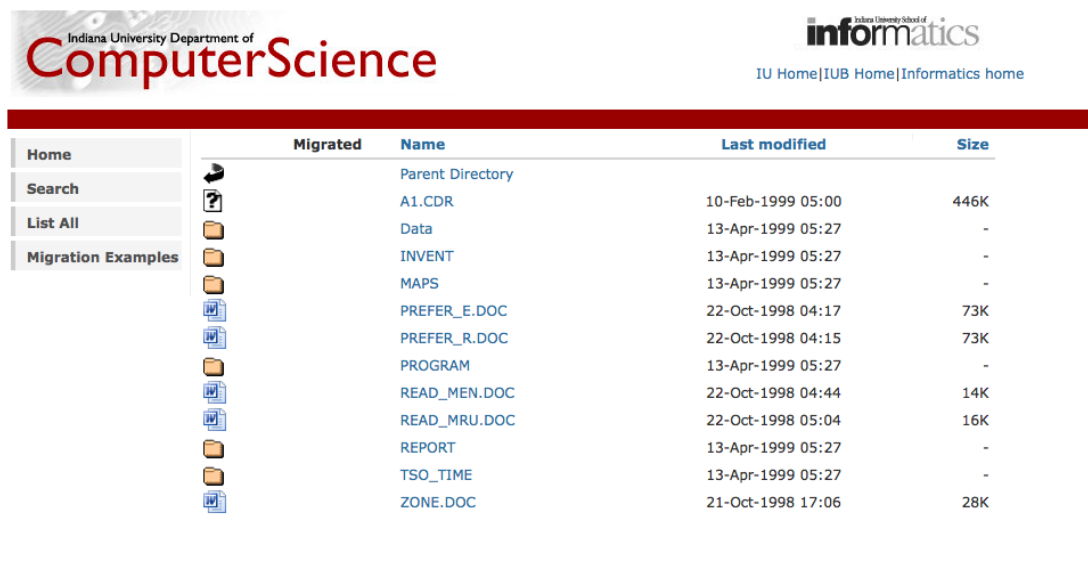
1. [Download](#) : [Browse](#)

[Raw Mets Record](#)

Figure 4.8: Metadata for a collection item.

A user clicking on “Browse” is directed to a simple Apache-style listing of the contents of the ISO image, reconstructed by reading the directory structure of the filesystem using `libiso9660`. In the top-level directory for this example, a number of Microsoft Word documents await migration. A virtue of this view is that the user can browse the original filesystem without loss of context,

while simultaneously retaining immediate access to artifacts (migrations, metadata, and supporting annotations) that are stored independent of the ISO image.



The screenshot displays a web interface for browsing within an ISO image. The header includes the Indiana University Department of Computer Science logo and the Informatics logo. The main content area shows a table of migrated files with the following columns: Migrated, Name, Last modified, and Size. The table lists various files and directories, including Parent Directory, A1.CDR, Data, INVENT, MAPS, PREFER_E.DOC, PREFER_R.DOC, PROGRAM, READ_MEN.DOC, READ_MRU.DOC, REPORT, TSO_TIME, and ZONE.DOC.

Migrated	Name	Last modified	Size
	Parent Directory		
	A1.CDR	10-Feb-1999 05:00	446K
	Data	13-Apr-1999 05:27	-
	INVENT	13-Apr-1999 05:27	-
	MAPS	13-Apr-1999 05:27	-
	PREFER_E.DOC	22-Oct-1998 04:17	73K
	PREFER_R.DOC	22-Oct-1998 04:15	73K
	PROGRAM	13-Apr-1999 05:27	-
	READ_MEN.DOC	22-Oct-1998 04:44	14K
	READ_MRU.DOC	22-Oct-1998 05:04	16K
	REPORT	13-Apr-1999 05:27	-
	TSO_TIME	13-Apr-1999 05:27	-
	ZONE.DOC	21-Oct-1998 17:06	28K

Figure 4.9: Browsing within an ISO image.

Migration

In this chapter I describe the creation of a series of automated migration systems designed to simplify web-based access to collections containing a large range of legacy document and media formats. I analyze performance and document rendition quality using server-side migration software built from open source components.

In the context presented here, digital object migration is one path selected to ensure usability of digital objects in the future. The fundamental design goal of this architecture, *ongoing accessibility*, depends implicitly on avoiding a single points of failure when planning for future use. As such, migration is applied to objects for which the risk of information loss is lowest using currently available software technologies: office documents, images, and video and audio formats. Additionally, I explore migration strategies to extract browsable information from complex scientific and database-supported formats.

A primary goal of this migration implementation has been to enable sustained local and remote access while reducing dependency on legacy supporting applications and operating systems. It is designed to allow searching and browsing of the original files within their original contexts utilizing binary images of the original media. The architecture uses static and dynamic file migration

to enhance collection browsing, and emulation to support both the use of legacy programs to access data and long-term preservation of the migration software. A typical user browsing an online collection of legacy digital materials for documents of interest is likely to encounter objects in numerous formats which are not “browser-friendly”. For example, a user may find a Microsoft Word document that appears to be relevant, but is on a system that does not have a version of Microsoft Office (or the correct version of or viewer for Office documents). Providing PDF renditions of such documents alongside the originals – which may also be downloaded – can dramatically improve the user experience. In this chapter I examine other such paths to support browsing, including conversion of image and video formats, on-the-fly extraction of files and data from compressed containers, and XMLized renditions of tabular and database formats.

This remainder of the chapter describes both architectural considerations in designing the system and an in-depth analysis of file migration using data gathered from testing software on the formerly described GPO CD-ROMs and DVDs.

5.1 Introduction

A fundamental design goal in designing this system was to reliably support *migration on request*, providing conversion to modern (and web friendly) formats during each access event by a user, and collecting performance data on the ability of the system to service large quantities of requests. Additionally, I examined methods to build persistence into the system, attempting alternate migration paths and providing “soft” failures.

In order to gather relevant statistics on both performance and storage requirements, the software can operate in “static” (pre-cached migrations) and “dynamic” (on-request) modes.¹ In its

¹The strategy of migration on request uses original documents as the input for migration tools that may change over

current form, the software additionally employs emulation – in the form of script-supported commercial virtualization tools, to ensure continued access to documents where migration is infeasible or likely to result in information loss. The virtualization component is discussed in detail in the next chapter.

The test workload consists of thousands of document collections distributed by the United States Government Printing Office (GPO) through the Federal Depository Library Program (FDLP) over a 20 year period [27]. The filesystem images used in this migration analysis were extracted directly from 4199 CD-ROM and DVD disks held in the GPO FDLP and international collections at the Indiana University Libraries [12]. They consist of millions of individual files including fundamental data on economics, environment, population, and life and physical sciences. Accessing many of these items requires installation of proprietary and increasingly obsolete software. For example, a sample of 1,885 discs contains 19,089 Lotus 1-2-3 files for which support has been discontinued in Office 2007 [26]. The FDLP collections are an attractive research model for preservation because of the diversity of the formats used and the time span over which they were created. Furthermore, the number and size of the collections mandates the use of automated preservation techniques which, as I shall describe, stress the available tools.

The rest of the chapter is organized as described below. I provide an overview of the system architecture and current implementation in Section 5.2. I discuss data about the test collection, including the set of file types and number of objects of each file type, and data about migration including associated space and time requirements and information about the failure modes of the migration tools used in Section 5.3.

time [67].

5.2 System Overview

The initial prototype implementation combines the web interface described in the previous chapter with a logical file system model. The implementation is based upon the Django framework written in Python [22]. The primary reason for utilizing this framework is that it naturally supports a file system organization; a Django application is organized around a set of “path rules” which are used match against the path in a URL request. A selected rule is directly mapped to executed code. This code may access static files or generate dynamic content. Django provides an attractive model that maintains a strong separation of content and functional elements. A core advantage of this framework is an object-relational mapper that allows data models to be defined entirely in Python. Existing database schemas can be automatically imported into the framework and accessed via an internal API.

This reference implementation demonstrates the feasibility of using off-the-shelf, open-source tools to provide web-based file access, dynamic and batch-mode file migration, user authentication, permission profiles, metadata modification and storage, and related administrative tasks. Django provides an attractive model for site creation that maintains a strong separation of content and functional elements. A core advantage of this framework is an object-relational mapper that allows data models to be defined entirely in Python. Existing database schemas can be automatically imported into the framework and accessed via an internal API.

These facilities can be used to generate an appropriate object model for a preexisting MySQL database associated with the FDLP materials. In this implementation, models are generated for site administration, user authentication, access restrictions based on IP, and UNIX-style user-group associations and group permissions. Finally, models to support both pre-defined and dynamically created file “white outs” within the archive are instantiated.

The web interface includes a search facility that allows location of documents using the CQL query syntax [19]. Conformance to the SRU protocol ensures compatibility with existing systems [80]. Additionally, bibliographic records for each of the FDLP ISO images allow search of the entire archive collection via a database query.

The screenshot displays the SUDOC Virtualization Project web interface. At the top, a navigation bar shows the user is accessing from IP 129.79.244.199, currently logged in as 'as', with links for Recent Changes, Logout, and Login. A left sidebar contains navigation links: Search Site (with a Go button), Home Page, Contacts, Database, Search, Help, and Preferences. Below these is an RSS feed icon and a section for file masking settings, which are currently disabled for executable, installer, library, driver, and other files. The main content area is titled 'Index of image_30000087742411/30000087742411_1/'. It features a table with columns for Migrate, Name, Last Modified, and Size. The table lists several files, including PowerPoint presentations, SST files, an HDF file, and HTML files.

Migrate	Name	Last Modified	Size
	Banzon.Laquila.8.26.02.ppt	13-Jan-2003 14:38	43M
-	D1.sst	31-Dec-1969 18:00	-
	Evans.Laquila.8.26.02.ppt	13-Jan-2003 14:37	27M
-	MO36FW26.chlor_a_2.ADD2001353.004.2002213211116.hdf	15-Aug-2002 08:09	570k
-	N1.sst	31-Dec-1969 18:00	-
-	P1	31-Dec-1969 18:00	-
-	QualityFiltering.html	15-Aug-2002 12:36	6k
-	ReadMeDisk2.html	29-Jan-2003 17:37	8k

Figure 5.1: User navigation of objects within an ISO image.

On completing a query, the user is presented with a view of the relevant items from the “virtual” file system constructed from the individual ISO images. Additionally, the user is presented with relevant statistics on the search, including the number of items restricted based on the user’s privilege level and location. On selecting an item, users with appropriate permissions are provided the opportunity to update the archival metadata associated with that item. A sample view is provided in Figure 5.1.

Navigating within a given ISO image, the user interacts with a hierarchical file-folder interface

similar to the default presentation of static directory trees by Apache. However, this view is moderated according to the existence of “white out” information, user access permissions, and availability of migrated files. For example, a directory containing an outdated version of Adobe Acrobat may not be displayed to the user. A series of files with location-based restrictions likewise may not be displayed to a user on an inappropriate subnet, or within a restricted IP range. Those files that are candidates for migration may be displayed with an additional icon linked to the migrated file. A simple AJAX routine is used to poll the readiness of migrated items for those which are handled on-demand.

Server-side modules and utilities for file migration may operate independently of the web-based interface as daemons in full-batch mode or be triggered by user navigation. A model within the Django framework specifies those file types that are candidates for migration and whether the migration for a type should occur synchronously - for each file on request from the server - or asynchronously. Asynchronous migration requests are triggered the first time a user navigates to the top-level directory for a particular ISO image. All files with types corresponding to asynchronous migration requests within that image are migrated while the user continues to navigate the archive (or alternatively, pre-computed through batch processing).

5.3 Document Migration

In this section, I describe an approach to migrating legacy files into modern formats and extensive experimental results analyzing the success rate of our tools as well as the space and time cost of migration. The goal of these migration procedures is to enable users to explore the available collections in order to identify documents and data for further study. I convert formats that are not

readily accessed within a web browser - including office documents, databases, and older multimedia - into browser accessible formats including HTML, PDF, and Flash Video. I also examine a customized approach to an important technical problem relating to document preservation – the reliable identification of file formats.

The migration model used is *migration on request*, which translates from the original file format to modern format in response to user requests. Migration performance can be enhanced by “caching” pre-converted files; however, the conversion path is always from the original file to modern rendition.

The migration tools we analyze are all open source. For example, we use OpenOffice to convert Microsoft Word, PowerPoint, and Corel WordPerfect files to PDF [83]. Similarly, we use open source libraries to generate HTML renditions of DBase and Microsoft Access databases. An alternative technology which we intend to explore in subsequent work is the use of Microsoft Office on a virtual machine as a translation server; there are well known risks to this approach [74].

The remainder of this section is organized as follows. I begin with a review of file format identification and validation. I describe the migration tools and their average time cost and success rate. Finally, I examine both the relative merits of migration and provide a deeper analysis of performance issues including the space requirements for caching precomputed migrated renditions.

Format Identification

Reliable automated identification of file formats is a key technical issue with document preservation. There are projects focused upon cataloging and identifying file formats, but there are no solutions which are immediately applicable for wide-coverage migration profiles that match the tasks performed here [90, 35, 33].

The migration experiments described here were performed at a relatively stage early in this research. Later implementations of the system used the Shared MIME-type database for high-quality, high-performance identification. For the offline batch migration tasks discussed here, file format identification was performed using the UNIX `file` utility and suffix matching [31]. (A similar approach is used in the DROID system [7]). The major file types identified using these techniques and their frequencies are shown in Figure 5.2. In all cases, a given file or set of files (located via an absolute path or directory walk, as appropriate) is identified via file extension match. Each item is then tested with the file utility to determine whether or not it is a suitable candidate for conversion. As an example, many early files with the .DOC extension are not Microsoft Word Documents, but rather simple ASCII text files.

It is important to note that there is no reliable, fully automated way to identify file formats. For example, dBase index files cannot be identified by content, though their suffix (`.idx`) in the context of related database files is a strong indicator. It was necessary to “tune” the database used by `file` in order to eliminate obvious false matches and to develop suffix matching in the context of the collections we are studying. While tuning the identification heuristics was a necessary step, it was not particularly time consuming. Ultimately this hybrid format identification module proved sufficient for the current set of experiments, in spite of known limitations. Using the data extracted from file identification, we selected the migration paths in Table 5.1 for further study.

Those file formats selected for migration comprise nearly 25% of the 3.9 million files within the complete archive. Of the remaining documents, approximately 33% (915,138 HTML and 448,716 additional ASCII) required no migration. 18,461 files could not be read due to permissions errors or corrupt images.

653,287 files in the archive were identified as binary data without a further application reference. We developed a simple automated test to determine whether an individual file’s contents

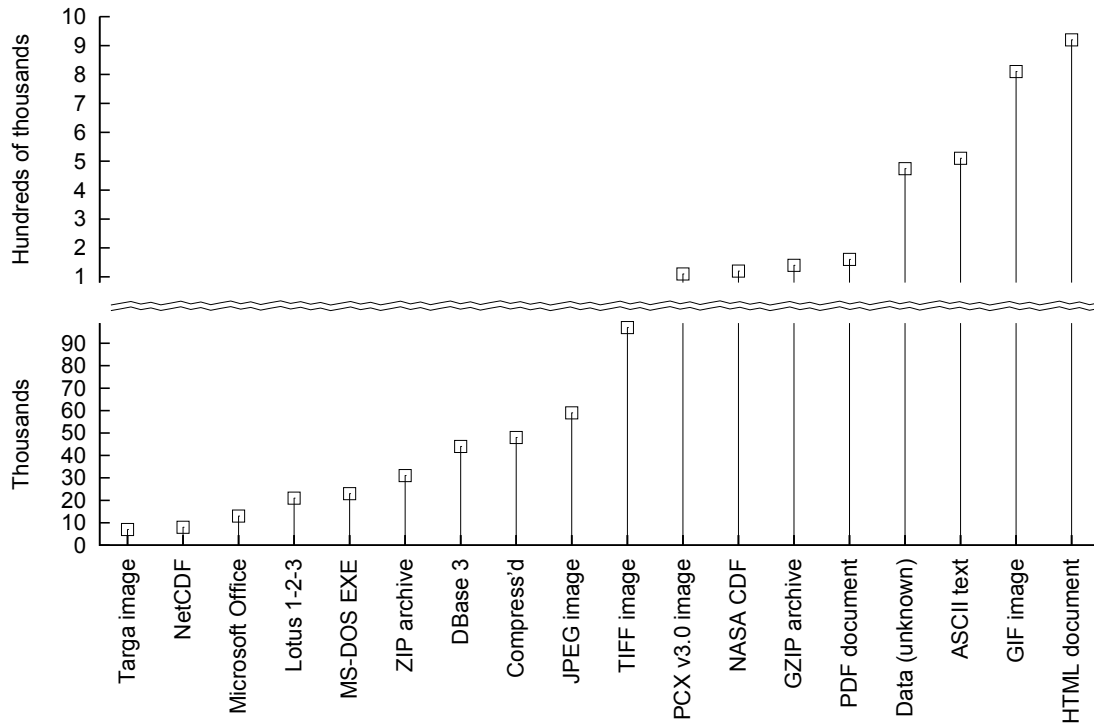


Figure 5.2: Top 18 file distributions by type.

consisted of greater than 70% ASCII text, and found 57,454 of these files fit this criterion. Of the remaining 596,265 “true” binary files, 122,079 were NASA CDF files already accounted for in the migration scheme. 88,117 of the remaining files were easily identified by common extension and automated scripted tests as geospatial (GIS) data. An additional 86,671 GIS-derived binary files with no extension were located in related directories. Approximately 30 other binary formats were observed – including various MS-DOS, Windows and UNIX executables, font files, and database support documents – each comprised of between 1,000 and 10,000 files. Several hundred older or more esoteric formats with fewer than 100 files were also identified.

Migration Tools

All migration tasks are handled with readily available open-source tools. A Python wrapper implementing a standard UNIX-style daemonization routine along with a high-precision timing library allows batch-mode processing of all relevant file types. Conversion of Microsoft Word and PowerPoint documents is performed via an instance of OpenOffice 2.2 running in “headless” mode, bridged through Py-UNO to a Python script handling associated path construction and cleanup tasks. All image conversions are generated by the Python Imaging Library. Microsoft Access and DBase III/III+/IV files are converted with Python modules obtained from SourceForge that extract relevant file data directly from the known binary format. Spreadsheet data from Lotus 1-2-3 is extracted to a structured XML file via the Gnumeric sconvert utility. Spreadsheet data from Microsoft Excel (through Office 2003) is extracted to a structured XML file via the xlrd Python module. Media objects, including MPEG encoded video clips and .VOB files extracted from DVDs, are re-encoded in a web-suitable resolution (320x240) Flash Video using the ffmpeg utility.

Three basic profiles were created for translation of items within the archive:

- Batch conversion of all applicable file types within the entire archive, performed at appropriate intervals and kept in store. For very large archives, storage cost associated with this

	Suffix	Number in Archive	Destination
Image formats	.gif .bmp .pcx .tif .png	697624	JPG
NASA CDF	.cdf	122079	ASCII text, XML
DBase III, IV	.dbf	35070	XML / HTML
Lotus 1-2-3	.123 .wk1 .wks	19089	XML / HTML
MS Excel	.xls	12279	XML / HTML
MS Access	.mdb	2805	XML / HTML
MS Word 6, 95, 97	.doc .wri	1804	Adobe PDF
Video formats	.avi .swf .mov .mpg .vob	1224	Flash Video (.flv)
Corel WordPerfect	.wp4(5,6) .wp	1104	Adobe PDF
MS PowerPoint	.ppt	380	Adobe PDF

Table 5.1: Conversion candidates

may be undesirable or prohibitive, although the continued rapid decrease in cost of storage mitigates this.

- Batch conversion of archival items based on access profiles. Requests for certain areas of the archive can trigger these conversions. Advantages include storage cost, suitable for rarely accessed archives.
- Conversion of data items just-in-time, on request. Good candidates for this approach include image files which may be rapidly translated and smaller non-binary or open binary formats. This method may also be used to partially convert substantially larger items, providing pre-view snapshots.

More complex translation paths are possible. Those documents in proprietary binary formats for which no freely available tools exist may still be converted efficiently without requiring the use of a fully interactive emulated environment (for example, scripting of DOS-based tools for file processing as batch jobs under Wine). As these experiments focus on on-demand migration for document browsing, a ‘simple’ strategy suffices, where objects are retrieved directly and migrated via a direct mapping to the appropriate tool. In systems where migration is performed as a long-term accessibility strategy, migrations may be triggered by an automated request from a migration registry, and applied to objects for which the contained files meet specific criteria. A basic outline of such a service is given in Figure 5.3. The complex migration path described is an approximation of prototype functionality described in the koLibRI documentation [54].

At the heart of this problem lies the question of whether dynamic (e.g. on request) migration is an appropriate strategy. As noted above, storage costs continue to drop rapidly, while the processing time required to translate even moderately sized documents can be considerable. For many archives, the cost of added storage may be negligible compared to other costs associated

with archive maintenance. Performing batch migrations to allow instantaneous access to the full document collection can therefore have significant advantages.

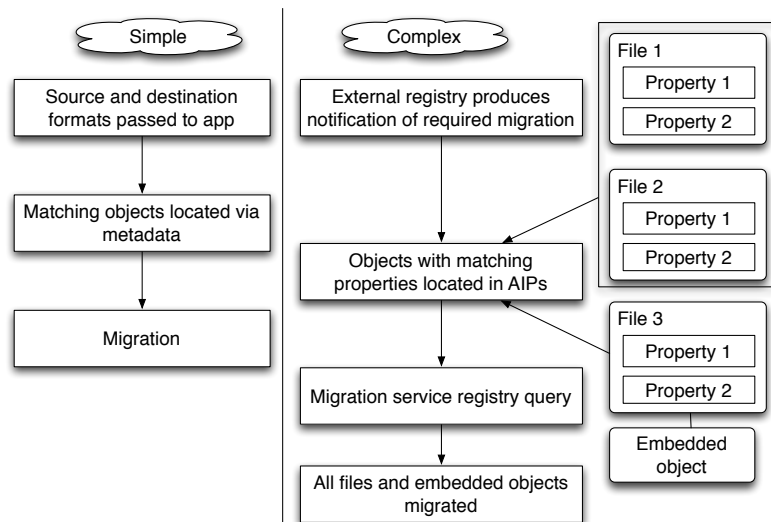


Figure 5.3: Migration strategies.

I describe the techniques used to translate each of the identified file types and evaluate two distinct criteria for those types: the percentage of documents that can be successfully translated, and the average translation time required. A full evaluation of successful migration would necessarily involve multiple independent assessments of the migrated document contents as compared to our original “gold standard” archival objects. In these experiments, a migration is marked “successful” if a non-empty migrated object is produced without an error generated by the translation tool. This criterion for success is obviously not suitable for storage of migrated objects in an archival context. Given that the intended usage profile here is dynamic (on request) generation during browsing of the archive via a website, it does not introduce significant additional risk. As failed migrations are identified and improved migration tools become available, on-demand migrated objects can be regenerated from the original bitstreams. The results of the migration tasks are summarized in

Table 5.2.

These statistics are provided to illustrate the practical issues that arose in this migration strategy rather than provide a complete view of corrupt, damaged, or otherwise inaccessible items within the archive. As discussed in the previous section, a set of damaged images - notably those for which directory structures could be read but from which no files could be extracted - was not included in the conversion process. These numbers therefore provide an approximation of real-world operation on these tools when using undamaged ISO-9660 images. Validation of the media can be difficult, however. The standard UNIX *isovfy* tool, part of the *cdtools* package, generates basic statistics on image integrity. These include the information on the tool used to produce the ISO, extension (Rock Ridge, Joliet) characteristics, and errors such as zero-length files for which extents have been written. In this set of images, 7 were identified with one or more null extents, while over 200 exhibited errors related to the use of Rock Ridge extensions. In many of these cases, the errors resulted from versioning issues with the extensions and had no apparent effect on file access. In theory, these ISO images could be reconstructed to fully conform to the current standard with no loss of information. As the contents of ISOs can be appropriately matched using direct filesystem walks, this may be a viable strategy for long-term archival storage.

	Successful conversions	Failures	Time/File
Image formats	692118	2908	< 1s
NASA CDF	122076	3	3s
DBase III, IV	34352	718	< 1s
Lotus 1-2-3	19086	3	< 1s
MS Excel	12268	11	< 1s
MS Access	2084	1	< 1s
MS Word 6, 95, 97	1267	99	7s
Video formats	1176	48	6s
Corel Wordperfect	1099	5	6s
MS PowerPoint	374	6	5s

Table 5.2: Conversion success rates

Microsoft Word, PowerPoint, and Legacy Word Processing Formats

Migration of Microsoft Office documents in an automated environment presents a series of challenges. Microsoft specifically recommends against the use of server-side automation with the Office suite of applications, primarily for stability and security concerns. Additionally, alterations to the proprietary Microsoft Compound Document File Format, along with changes in the Windows operating system, have led to modern versions of the Office suite which will no longer reliably read all documents created with previous Office products. While Microsoft provides additional tools for the migration of such documents (at the time of writing, a conversion utility for migration of all documents from Office 97 through Office 2007 is available), there are similarly subject to server-side scripting concerns.

The OpenOffice document suite and associated APIs were used in the interpretation and migration of existing Microsoft Word and legacy WordPerfect documents. OpenOffice is capable of reading the proprietary Microsoft format, can be run in a “headless” environment suitable to execution in daemonized mode on a server, and supports a number of language bindings through the Universal Network Objects package that make scripting of conversion processes possible. While OpenOffice continues to suffer from a number of stability problems in the context of server-side automation, we were able to mitigate the effects of these through the use of a watchdog timer tuned to terminate processes after a threshold period. The GPO materials examined contain 494 files identified as productions of early (MS-DOS) versions of Microsoft Word; these cannot be handled directly by OpenOffice, and no conversion attempt was made on them. Of the remaining 1816 Word documents, approximately 450 were discarded prior to conversion due to errors in disk permissions, lack of data content, or other I/O issues related to access to the original file. Of those 99 conversion failures observed, the majority were on attempts to read and convert ASCII files containing some small percentage of non-ASCII data, and identified by the UNIX `file` command as

data. Statistics for the various office formats are provided in Table 5.2.

Spreadsheet and Database Formats

The `ssconvert` program (part of the Gnumeric spreadsheet package) was used to interpret and migrate Lotus 1-2-3 documents, along with a variety of additional open-source libraries to handle Excel, Access, and DBase documents [38]. Data extraction from multiple worksheets or record format descriptions into individual HTML tables (or appropriately defined XML documents) is relatively straightforward. However, conversion may involve some loss of formula data or OLE inclusions.

For the Access and DBase formats, each migrated document represents a preview of the table structure, rather than a structured dump of all data within the original file, yielding significantly smaller sizes in the migrated collection. The purpose of this truncated migration is to enable web browsing without excessive download and rendering times.

The relatively large failure percentage on `.dbf` files is due primarily to a selection of files with correct header information but which were otherwise empty or contained malformed tables. These files were predominantly clustered on a small (less than half a dozen) number of media images which may be corrupt.

Video and Image Formats

Candidate multimedia formats for conversion were selected based on frequency of occurrence. The era from which this collection dates precludes the appearance of a number of currently popular formats, including Windows Media video and MP3 audio files. FFmpeg with an appropriate version of the libavcodec was used to transcode all video sources into a Flash video format suitable for embedding [30]. All videos were re-encoded at a fixed bitrate to a resolution of 320x240, typically

yielding a factor of 10 compression. All image data files were migrated to the JPEG format using the Python Imaging Library. Of the failures, a small number (73) were PCX files with non-v3.0 encoding. 2835 TIFF images with non-standard encoding or otherwise damaged data structures failed conversion. Finally, 5506 GIF images with correct heading information but containing no image data were disregarded.

Common Data Format

The FDLP collection contains a large number of documents from NASA encoded in the Common Data Format (CDF), a self-describing binary format used to store scalar and multidimensional data [13]. In addition to a web client for format translation, NASA provides source code for a number of tools capable of interpreting this format. In these experiments “skeleton” was created for each CDF document in the archive, consisting of the format-internal metadata along with a formatted list of variable identifiers and document data. The *SkeletonTable* script in the CDF utilities distribution creates ASCII representations which may contain data values for only a subset of the document-internal variables, resulting in significantly smaller migrated files suitable for preview via the related web application.

Performance Issues

While the migration times for most files as indicated in Table 5.2 are relatively small, they may be too large in certain cases (e.g. Microsoft Word documents) to provide an adequate browsing experience. As discussed in this section, these averages can be misleading due to startup costs associated with translating a single file that are obscured by this metric. In the following we provide further data about the *latencies* associated with migrating files on request. Table 5.3 illustrates

the space required to store precomputed migrated renditions for the various formats in the GPO-derived test collections. The entire archive occupies roughly 1.1TB on disk.

OpenOffice, which was used to convert various Microsoft Word, Powerpoint, and Corel WordPerfect formats, has significant performance overheads when used to convert single files. For example, in our trials Microsoft Word documents were converted in batches of 10, with a new child process running an instance of OpenOffice used for each batch. This operational profile was selected to overcome runtime and memory usage issues associated with OpenOffice 2.2. On our reference hardware running Red Hat Enterprise Linux the startup time associated with each instance of OpenOffice was approximately 3 seconds. The storage requirements for preconverted versions of all documents migrated with OpenOffice are small, totaling just over half a gigabyte.

	Number Converted	Original Size		Converted Size	
		Total	Mean	Total	Mean
Image formats	692118	50676MB	75KB	118074MB	175KB
NASA CDF	122076	148543MB	1247KB	9121MB	77KB
DBase III,IV	34352	65410MB	1991KB	1360MB	41KB
Lotus 1-2-3	19086	617MB	33KB	2142MB	115KB
MS Excel	12268	1038MB	84KB	3698MB	308KB
MS Access	2084	12241MB	6018KB	331KB	< 1KB
MS Word 6, 95, 97	1267	405MB	330KB	196MB	160KB
Video Formats	1176	7856MB	18MB	978MB	2308KB
Corel WordPerfect	1099	179MB	167KB	227MB	212KB
MS PowerPoint	374	199MB	800KB	132MB	53KB

Table 5.3: Conversion resource usage

In contrast, the various image formats have high storage requirements (both in original and migrated rendition) and low latency conversion cost. For these formats, synchronous migration on request appears to be both viable and desirable.

In handling formats such as NASA CDF, DBase, Lotus, and Excel, our migrated rendition contains only a subset of the original data (recall our objective to enable browsing) and hence the

storage costs for migrated renditions are small. Caching may be reasonable even with modest migration latency under such circumstances.

Finally, video formats are excellent candidates for caching, as we require only low resolution renditions to support browsing and the quality of these renditions can be improved with more computationally intensive codecs. As with all files in this archive, the user may elect to retrieve the original if needed.

The majority of the tools and automation scripts used in this application were tuned to take an optimistic approach to conversion, selecting all those files that matched either a UNIX `file` identification pattern or a simple extension, and discarding only those files for which there was a clear mismatch. Additionally, a simple timeout was used to kill off processes exhibiting blocking behavior (or otherwise excessively using CPU or disk for long periods). There are a number of failure conditions for which this simple approach is insufficient given the behavior of these tools.

These results represent the operation of tools modified over many iterations to provide acceptable results in an automated environment suitable for server-side operation. Binary file types, in particular, present a challenge in scripted settings where care must be taken to differentiate between programmatic exceptions related to the tool or I/O subsystem, and those which represent legitimate failures to convert a non-corrupt file. The relatively low failure percentages presented in the tables above, along with per-file conversion times suitable for on-the-fly (as opposed to batch mode) migration should not be therefore be interpreted as a “solution”, but rather as a strong indication that the problem of migration can be readily attacked with off-the-shelf tools.

Future work may include tunable profiles with timeouts optimized for file type and distribution within the archive, custom codec sets for multimedia translation, and additional pre-migration tests for file integrity. In the context of the web application for which these migration profiles have been developed, a small percentage of failures is not unreasonable, as the user has access to the original

files and may pursue additional strategies to translate these into an appropriate modern format.

The techniques described here establish flexible migration profiles to handle formats with migration paths that vary significantly in terms of latency and throughput. Advantages of this approach include the ability to dynamically provide multiple migration paths for individual files and to maintain real-time access irrespective of file size.

6

Emulation

Emulation is frequently discussed as a failsafe preservation strategy for born-digital documents that depend on contemporaneous software for access [40, 68, 100, 101]. Yet little has been written about the contextual knowledge required to successfully use such software. That is, legacy digital objects – particularly executables and interactive content – frequently make implicit assumptions about both the hardware platform on which they will be accessed and the expertise of the user in preparing and using that environment. They may rely on the presence of legacy media access devices or drive hardware not available on modern machines, or assume that the user is familiar with command-line interfaces that were at one time standard. In this chapter, I discuss an approach to preserve necessary contextual information through scripts created during the preservation workflow designed to control the legacy environment.

I describe software designed to minimize dependence on this knowledge by offering automated configuration and execution of emulated environments. I demonstrate that even simple scripts can reduce impediments to casual search and use of the digital objects being preserved. This can help eliminate both a dependence on physical reference workstations at preservation institutions, and provide users accessing materials over the web with simplified, easy-to-use environments. The

implementation is tested on a sample set identified within the virtualized collection of GPO CD-ROM ISO images, selected for coverage of a wide range of executables, supporting software, and other legacy environmental requirements.

6.1 Emulation and Access

Emulation is a key strategy for ensuring long-term access to born-digital materials, yet it has a vulnerability that is rarely discussed its dependence upon original software and the corresponding assumption that future users will remember how to “drive.” Even a seemingly simple task such as installation of software can be a formidable impediment to use. I describe tools that will assist future users by automating tasks such as software installation that could otherwise compromise usability, as well as techniques to automate other common tasks, such as exporting data and generating reports using preserved software environments. The installation issues could be overcome by the alternative strategy of creating a custom emulation environment for each preserved object; however, space and software licensing concerns make this a relatively unattractive choice. Furthermore, there remain usability issues that can only be addressed at runtime.

The software described here provides users with access to legacy executables in automatically configured virtual machines using simple installation scripts. Scripts and relevant documentation are stored alongside the original digital objects and metadata. A primary goal in this work has been to address basic questions about preserving any contextual knowledge associated with legacy executables. I begin with simple automation of common installation and execution tasks. Additionally, I discuss “wrapping” older DOS-based and early Windows GUI applications to generate reports or export data. Automation allows users to quickly browse and obtain information from

preserved programs much as they would traditional static documents, without the need to learn arcane installation procedures, risk contamination of the host environment, or manually reconfigure the virtual machine.

I show that with a small amount of coding - less than 600 lines of C# code to guide the user, configure an emulation environment, and run associated install scripts, and less than 100 lines of Java to link browsing of a web archive to the local application - creating stable, repeatable environments for virtually any application type is straightforward. Typical object specific install scripts are small (1-10 lines of code).

The work assumes that emulation is necessary to preserve some materials, and that the underlying technology is sufficiently well understood that preservation of emulation environments is likely to remain viable. I specifically address automating access to materials within an emulation environment, preserving and sharing emulation environments, and the technology required to automate emulation.

Digital archives with a mandate for open access must inevitably implement strategies of this type. Modern users are accustomed to convenient access to archival materials over the Internet, but do not necessarily have the expertise to reconstruct the environments necessary to use these materials – a problem that has been discussed in the development of some preservation-aware emulation prototypes [107]. A survey of more than 4000 legacy CD-ROMs in the GPO collection at Indiana University identified more than 1900 unique binary executables distributed without source. These programs encode historically significant scientific, economic, legislative, environmental, and social data – data that is media-rich and frequently cannot be migrated or reprocessed into static documents without information loss.

Existing archival strategies and the software tools that support them have increasingly focused on considerations of future access - who will be looking for these materials, how they will find them,

and under what conditions they will be used. Many of the assumptions made in these strategies are predicated on the idea that the majority of these materials will be stored as traditional documents - frequently word processing, spreadsheet, database, or presentation formats. Executables present more significant access and search problems. Modern indexing schemes are based primarily on document metadata and, to a lesser extent, on document content. As the complexity digital objects increases, facilitating end-user access becomes as critical a preservation problem as retention of the metadata used to describe them. Assisted emulation schemes such as the one presented here are a useful component in systems addressing this issue.

The remainder of this chapter is organized as follows. I outline the approach, including basic layout of the client, server, and networked storage, and discuss the specific problems addressed by this software. I provide a detailed description of the operation of our software, including the adaptation of an existing online archive for both on-site and remote use. I describe preparation of the archive, including script generation for automated installation and data extraction. I examine existing emulation platforms and preservation-specific emulation approaches, and discuss some future directions for this and related work.

6.2 Approach

This set of trials focuses on improving user navigation of and access to programs located within a networked collection of CD-ROM ISO images presented as a single filesystem and accessible through a web interface (<http://www.cs.indiana.edu/svp/>), augmenting the access and migration profiles described in the previous two chapters. When a user clicks on a link to a virtual disk image for which an installation script is available, a Java applet executes a helper application on the users workstation taking the form of a “wizard” that informs the user and provides the option

of mounting the ISO image within a guest (Windows) virtual machine.

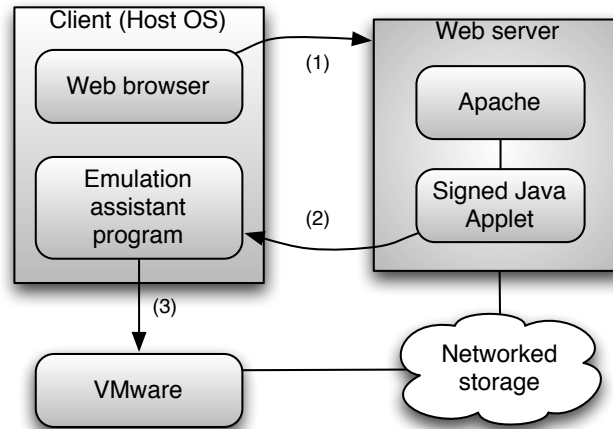


Figure 6.1: Automated launch and configuration of a VM

Figure 6.1 illustrates the basic interaction between a user with an appropriately configured workstation and a remote site with legacy digital materials held on networked storage. A user navigating an online archive via a web browser locates a resource by entering search terms and selecting a desired ISO image from the result list. The user may browse the contents of the image directory structure in the browser, but for those images containing executables, two links are provided for direct access: one which executes the local helper application to configure a virtual machine using the appropriate network resource (.iso file mounted as a “virtual” drive), and one which provides the option of downloading the entire image.

Installation scripts are stored alongside archival materials on logical volumes in a distributed, networked Andrew File System. Within each volume, .iso objects are stored in directories uniquely named according to a barcode assigned to that object. Installation scripts and additional metadata are stored alongside the CD-ROM images, as diagrammed in Figure 6.2. The software automatically polls a directory upon selection by a user to determine if an installation script is available.

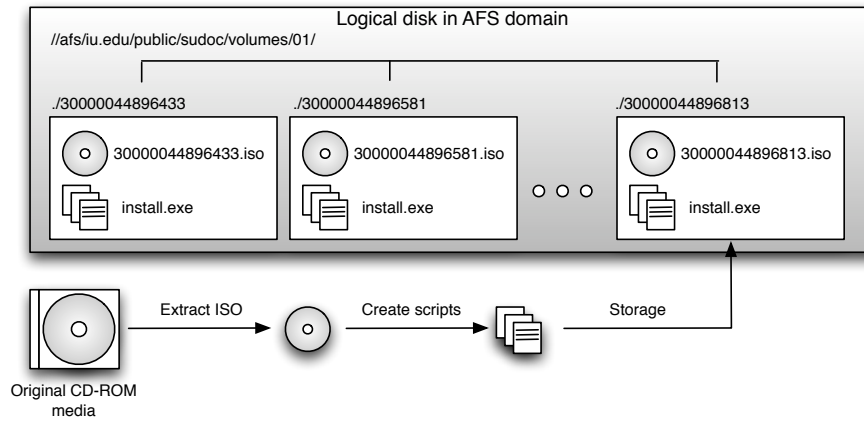


Figure 6.2: Organization of the archive

There are several advantages to this approach. ISO images of CD-ROMs are large (typically hundreds of megabytes), placing considerable overhead on the network and subjecting users on typical network connections to a lengthy wait. When large resources like `.iso` files are stored in a distributed, networked filesystem such as an AFS, a mount of the resource as a virtual drive in a local VM enables network transfer of only those parts of the image that are required. Multiple users or virtual machines can access the images simultaneously. Finally, distributed filesystems that support a global namespace improve the ability of institutions and researchers to share, compare, and verify both data items and the environments used to support them.

Automated virtual machine configuration and networked resource storage eliminate the need for libraries to maintain physical “reference workstations”, or hardware platforms with customized software environments to support access to legacy electronic materials. I focus in particular on executable software originally distributed on legacy media, including CD-ROMs and floppy disks. Given these use profiles, future users may find their experiences interacting with a particular collection using modern hardware is identical irrespective of their physical location or local software

environment. With minor modifications, this software can accommodate any VM supported by the emulation software and any form of media capable of being mounted as a virtual device in the VM.

When interacting with legacy executable software, the user may find that lack of knowledge about an outdated operating system - or the time required to acquire that knowledge - hinders their ability to effectively browse the available materials. This issue has already arisen for older Windows and MS-DOS environments. What percentage of modern users browsing an archive are familiar with editing DOS-era batch files, or could diagnose silent installation failures due to a hardware incompatibility? Careful documentation and operational metadata can mitigate the problem somewhat, but eventually these instructions become little better than incantations that the user must follow on faith, with little recourse should they fail.

Environmental integrity is another concern. A dedicated hardware workstation must either be preconfigured with every available piece of legacy software within an archive (a daunting task, not to mention problems associated with conflicting requirements and system security), or make available to the user installation and execution permissions that would lead to inevitable corruption or misuse of the environment. This can be overcome with the use of a virtual machine “snapshot” that is returned to a base configuration at the end of a browsing session, but the burden of manual installation remains with the user.

By providing a software-assisted method for automating interaction with the virtual machine, networked disk resources, and required software, this approach eliminates the problem of requiring extensive environmental knowledge from non-technical users. This software can be readily deployed on common workstations, and the associated Java applet code can be used to link user activity in a browser to execution of a local virtual machine. The automation scripts are linked to individual ISO images and assume the presence of a “clean” virtual machine snapshot, effectively “freezing” the environment. This eliminates concerns of future environmental integrity or software

changes. Additional scripts can be added to the archive as necessary without requiring alteration of the helper application.

6.3 Software

Overview

Our software includes four basic components: a small “wizard” application to provide the user with assistance in configuration and execution of an emulation environment linked to a networked archival resource, a signed Java applet to execute the wizard on the workstation when the user navigates to a resource of interest on a given archival website, scripts to support installation and execution of legacy applications held on media images within the archive, and scripts to support data extraction via export functions within the applications.

The user selects a resource by clicking an appropriate link to a signed Java applet (running server-side), which in turn executes a helper application installed on the local workstation. This application first prompts the user to select the desired emulation package (Figure 6.3). For this implementation, the software was tested with VMware Server 1.0.2 and VMware Workstation 6.0.4, to ensure compatibility of the API calls with a variety of common products.

The user is presented with a final confirmation of their selections: the location of the virtual machine - the application automatically searches for paths corresponding to valid virtual machines and presents a default choice in an intermediate dialog - and the path to an ISO image corresponding to their original selection on the web site. The application (shown in this stage in Figure 6.4) may also be run in a “standalone” mode, with the user selecting a disk image manually.

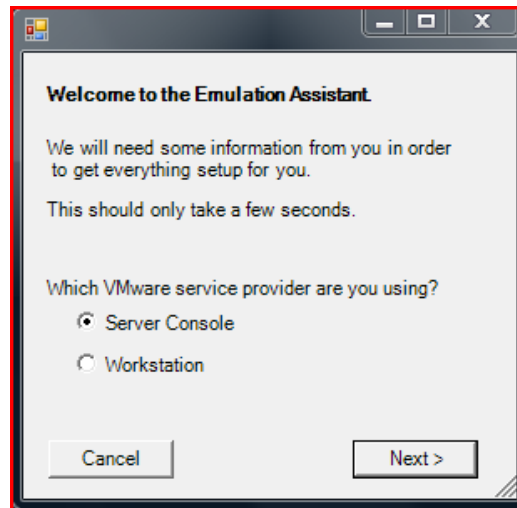


Figure 6.3: Emulation platform selection

Once these selections have been made, the software helper on the local workstation identifies the appropriate commands in the VMware configuration file, rewriting `ide1:0.filename` and `ide1:0.device` type to point to the appropriate ISO stored in the AFS. No further changes are required, although the application performs some basic tests to ensure that the user has not requested a missing resource. Finally, the application connects to the appropriate logical volume on the AFS to determine if any install scripts are available for the requested ISO. After a final user confirmation, the application starts the virtual machine from the existing snapshot, copies over any associated installation script, and executes it.

The ability to mount disk images as specific virtual devices, and beginning each installation in a “clean” virtual environment are critical components for successful access to many legacy executables. In the test archive, many of the associated installation procedures attempted to “update” the system with outdated library files. Manual installation attempts indicated that for a small but significant percentage of these installations, failure to install an older version of a software library broke the application. Additionally, many applications continued to expect to read resources (such

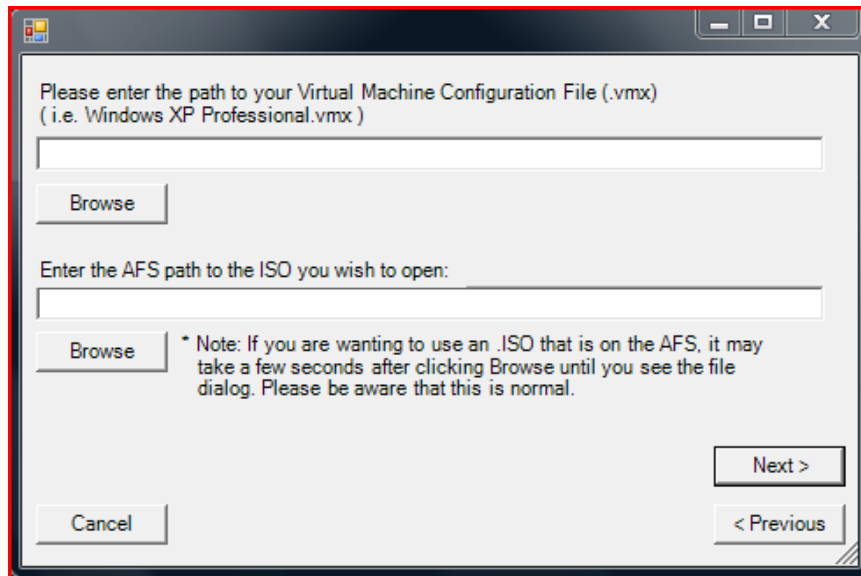


Figure 6.4: Client request for a networked resource

as records from database files, spreadsheet tables, or maps) from the original CD-ROM mount point even after installation.

Installation Scripts

Due to variability in installation routines and the fact that most legacy Windows executables are distributed as compiled binaries without source code, installation and configuration cannot be handled programmatically through the Windows API. Fortunately, a variety of high quality Windows GUI scripting languages exist for the express purpose of automating user-level tasks. These experiments utilized the AutoIt tool, which provides a simple BASIC-style syntax for automation of the GUI. The following code (taken from a sample collection developed as part of our testbed) demonstrates automation of a simple Windows 95-era setup procedure:

```
Run("D:\Win95\SETUP.EXE")
WinWaitActive("Welcome", "&Next >")
Send("!n")
WinWaitActive("Choose Destination Location", "&Next >")
Send("!n")
WinWaitActive("Select Program Folder",
    "&Program Folders:")
Send("!n")
WinWaitActive("Start Copying Files", "&Next >")
Send("!n")
WinWaitActive("Setup Complete", "Finish")
ControlClick("Setup Complete", "Finish", 1)
```

The installations scripts are frequently even simpler than this, particularly for DOS-era programs for which only one or two commands are necessary, or for which the installation assumes a default path without informing the user. For those DOS-based applications that required additional changes to configuration and initialization files, help documents were included to describe the process of preparing the scripted installation.

Certain installation procedures depend on more subtle factors related to the operating system. Early Windows application may make (or require the user to enact) changes to system settings, such as those related to available fonts, drivers, or languages. These requirements may be opaque to future users. For example, installation routines for some executables in the collection required the user to manually specify the maximum number of file handles available to the executable. In versions of Windows prior to 95, such changes were sometimes necessary to override system defaults set to conserve scarce resources. Scripted on-demand installation shields the end user from having to puzzle through historical technical requirements, without requiring the construction and maintenance of application-specific virtual machines.

A set of approximately 150 CD-ROM images containing candidate executables for scripting were selected from a list of more than 1900 custom executables identified in an original set of 4000

ISO disk images. 94 of the candidates had non-trivial installation procedures. A work-study student completed each installation in a “clean” Windows XP virtual machine, recording the necessary GUI actions, compatibility modifications, and (where necessary) alterations to the original installation procedure. Each action was recorded and encoded as an appropriate AutoIt command. 76 scripts were successfully completed. The remaining 18 exhibited compatibility issues with Windows XP. The successful scripts were subsequently compiled as executables and stored alongside the original archival materials in the networked Andrew File System disk array.

Scripted Data Extraction

Many of these legacy applications found on these CD-ROMs serve a single purpose to reformulate data stored in a common file format (such as dBase or Microsoft Excel) and present the resulting “views” to the user. In one common case, raw data such as economic or census figures are stored in dBase files upon which various joins and filters are performed based on selections made by a user in a Windows 3.1-era GUI application. The number of available views is often limited, and the application provides a facility for exporting a columnar view as a text file.

While it is possible to migrate the dBase data directly to a more modern format, this may result in loss of context; without the data manipulation performed in the GUI application, the intended visual structure of the data (frequently non-trivial) can be lost. Macro scripts such as those described in the previous section can provide a simple method for exporting structured information. In the example illustrated below, economic data published by the U.S. Census Bureau as a CD-ROM “County Business Patterns 1995-1996” is encoded in a series of legacy .dbf files that could readily be migrated to a more modern format using numerous open source and commercial software solutions. However, the data encoded in these files alone is not sufficient, as (for example) some field names are encoded numerically using the Standard Industrial Classification Code List,

and only mapped to their natural language counterparts in the final GUI display. A sample of such a display is provided in Figure 6.5.

County Business Patterns

Year: 1996 Geographic Area: CALIFORNIA County Name: Napa

SIC Code Description: --- TOTAL

SIC	Industry	Total Mid-March employees	Payroll (\$1,000) First quarter	Annual Payroll (\$1,000)	Total Establishments
----	TOTAL	41,993	261,813	1,123,665	3,417
07--	AGRICULTURAL SE	250-499	(D)	(D)	74
10--	MINING	0-19	(D)	(D)	6
15--	CONSTRUCTION	2,327	15,799	78,926	414
20--	MANUFACTURING	8,148	68,891	284,827	275
40--	TRANSPORTATION	1,291	10,311	43,586	106
50--	WHOLESALE TRADE	1,514	13,179	53,539	174
52--	RETAIL TRADE	9,214	31,111	141,140	773
60--	FINANCE, INSURA	2,278	19,154	75,708	308
70--	SERVICES	16,732	101,323	435,054	1,266
99--	UNCLASSIFIED ES	0-19	(D)	(D)	21

Number of establishments by employment-size class

SIC	1-4	5-9	10-19	20-49	50-99
----	1,855	732	422	281	75

Number of establishments by employment-size class

SIC	100-249	250-499	500-999	1000 or more
----	40	7	2	3

Next SIC

Secret.dbf Rdnly Rec 1137/1137 Ins

Figure 6.5: Reconstructed view of legacy database.

For researchers interested in collating or searching data intended to be viewed using these executables, assisted data extraction provides high-quality, low risk views into the original format while requiring minimal interaction with the emulated environment.

Linking

Because modern browsers operate in a “sandbox” designed to shield the user from malicious sites, we used a signed Java applet to link browsing activity with the local executable necessary to configure, boot, and perform required installations in a local virtual machine. Administrators

can deploy this solution (or a modified one using the available source) simply by adding an applet-based link which points to the desired resource within existing pages. In our implementation, these links were generated automatically in the following form:

```
<applet code="RunExecutable.class" archive="RunExecutable.jar"
        width="120" height="35">
<param name="exePath" value="C:\Legacy Emulation
        Assistant\Legacy Emulation Assistant.exe">
<param name="afsPath" value="\\afs\iu.edu\public\sudoc\
        volumes\04\30000038669341\30000038669341.iso">
</applet>
```

The Java applet reads two parameters: one corresponding to the path of the helper application on the users workstation, and one linking the networked resource. These can easily be customized; the applet exists only to provide a “trusted” link between the resource provider and the local workstation. Once executed, the local application itself can poll the system to determine what virtualization platforms are available.

Emulation Platforms

In order to be effectively deployed in a production preservation setting, an emulation platform must implement features not only for technical preservation activities, but also for administration, configuration, and maintenance. These facilities should be organized as management and automation APIs that can be used to normalize a virtual machine environment without concern for external factors such as where it has been deployed.

Emulation tools prepared by the digital preservation community remain largely primitive in terms of both technical and administrative features. Commercial products including VMware, Parallels, and CrossOver have set the standard for features we expect will be required in any large scale

preservation solution, including robust APIs for 3rd-party development, support for a wide variety of virtualized I/O, network, and hardware devices, and automatic methods for deployment.

VMware Workstation and VMware Server were used for this study, along with the VIX automation API due to its relative stability and widespread use. However, our solution uses fewer than 100 lines of C# code to provide the needed automation support (start and stop of the VM, device mounting, and file operations), and could readily be ported to a fully open source solution such as QEmu. While configuration of products such as QEmu is somewhat less “user friendly”, APIs such as libvirt provide functionality similar to that available in commercial packages.

Existing commercial and open source virtualization products are stable and mature. They are routinely used in mission-critical environments to provide users with flexible access to customized environments independent of the underlying hardware. These products demonstrate the immediate potential for improved access to digital archives, and suggest that existing emulation platforms already meet the primary requirements of many organizations within the digital preservation community.

6.4 Discussion

This chapter describes a software-assisted method to provide users with a simple and reliable method to install and execute legacy software in an emulated environment when browsing legacy archival materials. The implementation presented here has been extensively tested on the collection of legacy CD-ROM images available at <http://www.cs.indiana.edu/svp/>. Automation and data extraction scripts for the materials in the SVP archive can be accessed via an AFS client pointed to the publicly available archival space. Some of the VM automation code is included for reference in Appendix A.

These experiments are not intended to address issues of emulator preservation or selection of an emulation platform. The focus here is on demonstrating the feasibility of assisted emulation, particularly for archives with moderate budgets and technical expertise. The associated software is configured to interoperate with commercial virtualization products such as VMware that have mature APIs. The fundamental design use of an open source distributed, networked filesystem, executable scripts constructed using a GUI-level macro language, and a simple Java applet to provide browser support can be easily modified to support additional virtualization platforms and APIs.

This research describes simple software-assisted strategies to retain contextual information required to install, execute, and interact with legacy software. By automating both the process of configuring the virtual machine and of installing legacy software, basic impediments to access that current and future users might face when browsing these types of materials are removed. The assisted emulation procedures can be integrated into existing web-based access systems with relative ease and transparency, providing natural support for users who wish or need to access executable content in a legacy environment. The approach is flexible: it *depends* on only a single virtual machine that can be reconfigured and restored to an unchanged snapshot on demand, but can be extended to incorporate virtual machines of varying vintages and configurations. This reduces the storage and maintenance burden on the holding institution, while also allowing for improved access to specialty collections where required.

Conclusion

7.1 Discussion

This work presents legacy media transfer and handling technologies intended to assist archival institutions with legacy media holdings in the low-risk creation of virtual collections. I examine reliable and replicable methods for extraction of filesystems and files from ISO 9660 formatted CD-ROMs and FAT12 formatted floppy disks, demonstrating that common open source tools along with small amounts of customized code can be used to verify and maintain access to scientifically and historically important holdings. I show that many commonly used commercial tools fail to correctly extract or verify legacy filesystems – particularly where mastering problems are in effect – and explicate methods for avoiding hidden error in building archival-quality virtual images of these media.

I describe the numerous advantages gained via the construction of virtual collections, including the use of distributed storage and the advantages of reliable, secure network access to collections

which otherwise languish in physically isolated depository libraries. I show that the overhead incurred for access can be significantly lowered. Through the use of these methods and related software, libraries and archives can eliminate a dependence on physical workstations; organizations can share and enable crosswalks of metadata, verify holdings, and provide assurances of long-term access to materials with high scientific, historical, and cultural value.

The collection established here uses the interoperable METS metadata standard (along with MODS bibliographic sections) to reformulate existing MARC records and construct a searchable index for web-based access. The metadata schema was constructed in METS both for its relative ease of construction and handling and for its future extensibility. Although the existing implementation includes only limited rights and provenance data, the records produced can be quickly reconstructed with a high degree of automation to incorporate future design additions. Furthermore, while the MODS bibliographic metadata cannot be “round tripped” back to the original MARC records, there is no loss of information as the existing MARC information is retained for future use.

The “virtual filesystem” around which this collection is constructed provides a variety of significant benefits. The natural semantic links between originating media, extracted filesystems, objects within those filesystems, and the software and environments on which they depend are retained by virtue of both careful metadata design and the abstraction of the underlying storage hierarchy away from the view that is presented to those searching the collection.

By incorporating filesystem preservation, efficient metadata interpretation and transform, legacy file migration, and assisted emulation, the presented work avoids most of the common technical and procedural errors that can result in both loss of data and loss of contextual content through a single point of failure. Lossy or failed migrations can be reattempted from the original bitstream as technologies improve. The majority of executable materials in collections which require only a minimally configured virtual machine (or are distributed with their own legacy support software)

can be serviced with low storage and processing overhead.

The approach described here is a blueprint for creating high-quality, low-risk archives of CD-ROM and floppy disk images with an emphasis on access. I establish methods by which automated or scripted error detection can reduce risk in transfer of archival data from legacy media to low-cost, high-reliability redundant disk arrays. I demonstrate methods for processing a collection of legacy media and associated metadata and creating a complete web-access solution using freely available open source projects along with a few thousand lines of C, Perl and Python custom scripting.

7.2 Future Directions

Risk Management

Risk management is, in general terms, the process of identifying and mitigating risks associated with collections and file formats. These may include risks to binary interpretation during translation or rendering, the ability to reproduce necessary legacy environments, or other external factors.

There are three main sources of risk in handling digital materials. First, risk associated with the digital object itself. This may stem from documented and undocumented features of the format in which the object is encoded, changes to the format specification over time, and integrity of the bitstream. Second, risk associated with any tools used to create, migrated, or render the object. Finally, risk associated with the environment in which the object was created or where it will be rendered during future access events.

Specific questions to be asked about risk management include:

- Given a large (> 1M digital objects) collection drawn from existing archives or networks, what percentage of files can be reliably identified - where 'reliably' is taken to mean format, format

version, size, and interpretation of associated metadata and links to additional digital items - using automated techniques? For those items which cannot be identified using existing techniques, can heuristics or additional search criteria be identified to improve recognition?

- Can we identify and interpret all relevant features of the file format in question?
- How can we quantify (or improve on existing metrics) the level of risk associated with features of interest? To what degree can the identification and recording of these values be automated?

Answering these questions thoroughly depends in part on the complexity of the format. Certain legacy formats, such as Lotus 1-2-3, are relatively simple, well documented, and extensively supported by open source tools. Primary risks include formatting and formula calculation. For more complex document formats (including Microsoft Office), the investigation will focus on key risk factors common to a large percentage of documents found in real-world collections, including font usage, links to external resources, embedded media, revision histories, and additional metadata.

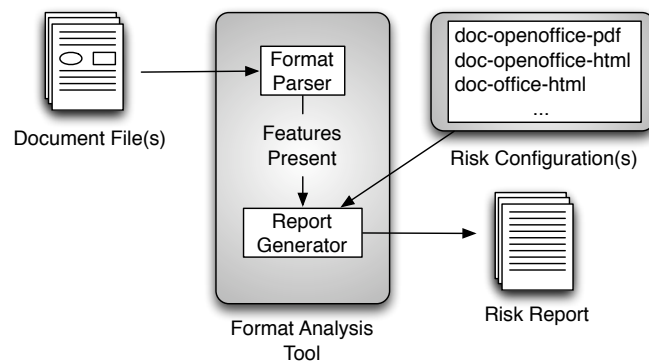


Figure 7.1: Format risk analysis.

Note that managing risk in the interpretation and potential migration of binary formats is different from the problem of *object validation*. The original and migrated renditions of a particular

document may conform perfectly to their respective format standards with the latter still exhibiting information loss. It is therefore not sufficient to validate objects *after* a transformation in an archival workflow; this process serves to mitigate a different type of risk.

In previous research, I have identified specific software tools for evaluating risk to particular formats, including the Shared MIME-Info database for reliable file-format identification, `libwv` for font extraction from Microsoft Word documents, standalone wrappers for Gnumeric conversion plugins targeted towards legacy spreadsheet formats, and the OpenOffice core libraries. Future experiments will emphasize *automated* analysis and verification of document composition wherever possible. A simplified outline of how such tools might operate is provided in 7.1.

A variety of tools exist to analyze proprietary formats at a low level. For example, Microsoft provides Win32 API support for access to Compound Document Properties for Office documents in the legacy Compound Document File Format.¹ In addition, published (although dated and incomplete) specifications for the individual Office document formats are available. These resources mean that for certain historically significant, wide-distribution formats we are not limited simply to comparison of rendered documents, or reliance on potentially incomplete feature extraction from external libraries. Rather, we can directly compare select features extracted by the legacy API with those available to migration tool candidates.

Preservation Quality Metrics

Collection and organization of risk-assessment data, along with improved software mechanisms to perform “deep” analysis of format-level risk, are required to support the ongoing needs of existing and future archives. Immediate benefits include improving the transparency of handling obsolescent formats, reducing reliance on proprietary rendering software, and identifying

¹<http://support.microsoft.com/kb/186898>

and reducing introduction of error during the preservation process.

In order for these benefits to be realized, such an approach must provide clear strategies and associated tools for the identification, extraction, and interpretation of *significant features* associated with commonly used document formats. By *significant features* I refer those aspects of the format or original rendition which contribute directly to the intended rendition of the object. For example, these may include formatting and typefaces in word processing documents, formula calculation in spreadsheets, or contextual dependencies such as links to external resources. Examples of basic rendering considerations can be found in recent literature [15].

Few formal metrics to evaluate success in document rendering or calculate risk in document handling exist, although recent work conducted as part of the Planets project has developed an XML language and tools to support characterization of digital objects based on features [5, 4]. The problem is compounded by the significant variance in file characteristics observed in the literally thousands of binary formats introduced (and obsoleted) over the past several decades. However, analysis of file distribution in existing archives and on the web indicates that the majority of these appear in a “long tail” of rarely used formats [114]. Conversely, a considerable percentage of business, governmental, and scientific data targeted for preservation is found in a core selection of document and office formats such as Microsoft Word, Microsoft Excel, Adobe PDF, and PostScript. By focusing on these formats it is possible to both narrow the scope of the problem while still addressing a large, heterogeneous collection of data.

It is critical to recognize that the application of various digital preservation practices to a collective body of knowledge is only relevant if it provides some guarantee that the contained works can be accessed efficiently *and without hidden error* by current and future users. As distributed and redundantly backed-up storage systems become the norm for archives, the major issues involved in ensuring this will no longer be media refresh and verification, but rather knowing that the available

tools used to interpret the digital objects limit the potential for introduction of error.

Reducing Rendering and Migration Risks

Both risks due to migration operations and errors encountered during rendering (both in original environments and of migrated documents) are endemic. Determining how to adequately address such risks in an automated or semi-automated fashion requires identifying and prioritizing specific technical issues with objects in legacy formats and with programs used to render them. In recent work carried out with Dr. Geoffrey Brown at Indiana University, I have examined one form of risk to legacy office documents - font rendering.

In this work we evaluated two data sets totaling approximately 230,000 Word documents to measure the difficulty of identifying referenced fonts given a database of font information. The primary identification technique we utilized was based upon name matching; however, we also evaluated the utility of the other font metrics recorded in Word documents. We had hoped to build a database of font information extracted from the fonts published by the major foundries; however, we found most foundries reluctant to provide the requested information and the cost of purchasing fonts not justified by this exploratory research. We resorted to combining information from a variety of sources including font files and font name information provided by several major foundries, extracted from published lists, and retrieved from foundry web pages. The font collections include fonts provided with various Mac OS X and MS Windows distributions, fonts distributed with applications such as Microsoft Office and WordPerfect, and fonts donated by Bitstream.

To determine the fonts used in Word documents, we wrote a custom application based upon the open source library libwv which is the basis for file import in Abiword and other applications (Lachowicz 2009). The libwv software provides a basic document processing function with application-specific callback functions. For our work, it was necessary only to provide a function

to track the fonts specified at the beginning of a text run and a function to process each character in a text run.

We established that the majority of digital documents obtained from a wide range of sources (up to 79%) can be rendered accurately using fonts appearing in modern desktop environments such as the combination of Microsoft Windows and Microsoft Office. With a small amount of additional work using information drawn from font foundries, performing family name matches for legacy fonts or commercial fonts for which distribution has ceased, we can expect to increase this coverage to 92%. This work is discussed in further detail in Appendix A.

7.3 Closing Comments

This work examines high-quality, low-risk techniques that can be used to build network accessible, preservation-aware collections from existing holdings of legacy media. The design and software technologies described in the previous chapters have been developed specifically to demonstrate that these types of collections can be constructed in a manner that is sustainable, efficient, and supports long-term access without placing undue technical burdens on holding institutions.

The methods and software described here provide collection browsing of both original and migrated materials, and enables execution of software artifacts in their contemporary environments utilizing emulation (virtualization). The technical components are organized in a modular fashion, with a “virtual filesystem” model providing contextual support and simplifying access. While publications often focus on the tradeoffs between emulation and migration [97, 96, 61], these technologies can be readily coordinated within a single model to improve both quality of access and

feasibility of long-term preservation. Both are considered acceptable document preservation strategies for government documents [23, 78]. The experiments carried out in this work provide a balance between instances where migration can be performed without loss, and those where migration paths are nonexistent or result in unacceptable or unknown losses. The use of primarily open source technologies ensures that these techniques can be replicated for similar collections at other institutions, and retention of all collection artifacts on a network-accessible filesystem provides an improved mechanism for sharing, replication, and audit of these materials.

A

Managing Risk

There is growing interest in the development and evaluation of risk management tools and techniques to collect, analyze, and disseminate quantitative information on extant and emerging risks to born-digital data [1, 85, 57]. While significant progress has been made, there are numerous open problems in this area. Imperfect and aging hardware, proprietary digital formats and supporting software, limited planning and formal risk management, and accelerating production of digital materials have highlighted limitations in existing infrastructure and in our understanding of what is required to maintain long-term access to legacy digital objects.

Experimental techniques to identify and control for information loss in document collections remain in their infancy. National and international preservation initiatives have focused largely on the development of preservation-specific frameworks, format registries, metadata production and handling, and reliable storage. While many reference extensive internal automation with respect to tasks such as format migration [10, 94, 111], few provide quantifiable data to support a reasonable level of scientific analysis and verification.

Published information on technical risks associated with digital documents remains relatively sparse. In part this is because problems of risk are endemic, and strategies to handle them are complex or depend on unproven (or unrealized) theoretical frameworks. Specific examples of risk include document formatting, inconsistencies in mathematical definitions used in different formats designed to support numerical data, and loss or corruption of metadata during access or modification.

Low-risk, scalable, and cost-effective document migration strategies based on open source technologies are required to provide support for the next generation of digital archives. Many of the risks associated with document migration – even for common procedures such as conversion of legacy Word documents to archival-quality PDFs – are not well addressed in current practice. Furthermore, recent strategies focusing on conversion to flexible migration targets such as XML [5] may not accurately capture the necessary behavior of complex digital objects that depend on legacy supporting software. Documented and undocumented revisions to format specifications and inconsistencies in implementation can produce significant information loss. As a simple example, many of the statistical functions in Microsoft Excel have been debugged or revised over time. While modern proprietary or open source spreadsheet software with the appropriate compatibility filters may be able to open legacy .xls files, the results may not match those seen by the creator.

In this Appendix, I review recent work undertaken with Geoffrey Brown at Indiana University investigating font use in legacy Microsoft Word documents [8]. This work explores risks to accurate rendition of such documents, describes the steps necessary to collect the necessary fonts (or font substitutes) from existing commercial and open font collections, and presents some statistics on likely rendering successes in similar collections.

A.1 A Case Study in Legacy Document Font Use

For millions of legacy documents, correct rendering depends upon resources such as fonts that are not generally embedded within the document structure. Yet there is significant risk of information loss due to missing or incorrectly substituted fonts. Large document collections depend on thousands of unique fonts not available on a common desktop workstation, which typically has between 100 and 200 fonts. Silent substitution of fonts performed by applications such as Microsoft

Office can yield poorly rendered documents and may result in significant information loss.

In this work we analyzed a collection of 230,000 Word documents to assess the difficulty of matching font requirements with a database of fonts. We described the identifying information contained in common font formats, font requirements stored in Word documents, the API provided by Windows to support font requests by applications, the documented substitution algorithms used by Windows when requested fonts are not available, and the ways in which support software might be used to control font substitution in a preservation environment.

The Font Substitution Problem

There are currently over 100,000 digital fonts in existence.¹ As a result, attempts to render a given document produced with specialized, commercial, or legacy fonts can result in missing *glyphs* (visible characters) or entire sections of content. Problematically, the reason for this may not always be clear because document editing suites such as Microsoft Office perform font substitution without warning.

Annoying font substitutions occur frequently. For example, symbols such as apostrophes and quotations are rendered with the “WP TypographicSymbol” font in WordPerfect Office 11. When these documents are migrated to Microsoft Word, this font dependence is preserved, and when the documents are rendered on machines without this font, these symbols become “A” and “@” as in: “in the AStrategies and Assessment@ Column”.²

The degree of information loss due to substitution depends both upon the importance of the glyphs substituted and their frequency. For example, corporate logos are often implemented with dedicated fonts containing a single glyph. There may be no substantive information loss from a

¹<http://www.microsoft.com/typography/FontSearchFAQ.msp>

²<http://www.wpuniverse.com/vb/showthread.php?thread-id=16756>

missing logo for most purposes. In contrast, substitution for mathematical symbols may result in total information loss. For example, our experimental data include 9 documents with program listings for the Texas Instruments TI-83 series calculators rendered with the `Ti83Pluspc` font which provides various mathematical symbols; these program listings are incomprehensible when rendered without this obscure font. This is illustrated in Figure A.1. Compounding the problem, Texas Instruments has published a variety of calculator fonts with different internal names and possibly incompatible glyphs.

```
Lbl A:ClrHome
If N<4:Goto Ø
CubicReg L1,L2:3→T:"aX3+bX2+cX+d"→Y1
Disp "aX3+bX2+cX+d"
```

Correctly Rendered

```
Lbl A:ClrHome
If N<4:Goto 0
CubicReg L ,L,:3→T:"aX3+bX2+cX+d"→Y
Disp "aX3+bX2+cX+d"
```

Default Substitution

Figure A.1: TI83plus font samples

Several documents in the collection were identified with barcodes rendered using the font “Barcode 3 of 9 by request”. When rendered with font substitutions, the bar codes of the numbers are replaced by Arabic numerals. Thus, the substitution preserved the numeric meaning, but not the functional ability to be scanned! Unfortunately, there are many barcode fonts and we were unable to find the required font; however, we were able to find a suitable substitute and to configure Office to accept our substitute as illustrated in Figure A.2.

It is difficult to create portable documents that will not suffer from font substitution when moved to another machine. Fonts are installed on a platform by the operating system, by applications, and by individuals. A user has no way to distinguish between standard fonts offered in an Office menu and those that have been installed by other applications. For example, ESRI

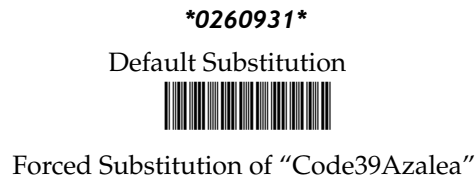


Figure A.2: Barcode Rendering

provides fonts with its various GIS applications. While these are relatively specialized applications that are unlikely to be present on most platforms, there is nothing preventing an Office user from utilizing the associated fonts. Furthermore, even the set of standard fonts changes over time – the fonts installed by Windows XP are not identical to those installed by Vista. Because of significant differences in the machine environments, there is a high probability that transferring a document between machines will result in missing fonts.

A solution to the problem of missing fonts for document preservation requires three components – identification of missing fonts, acquisition of the fonts or suitable substitutes, and configuring a suitable rendering environment including all fonts or their substitutes. This appendix covers work focusing on the problem of font identification. The issues discussed include extraction of font identification data from digital documents, the use of that information to match against a database of known font identifiers, and techniques for controlling font substitution.

The central analysis in this paper is based upon two large collections of Word documents – one described in [93] gathered using glossary queries to Google, and the second exclusively gathered from “.gov” sites. The second collection was created to test an hypothesis that government documents might use more restricted font sets; however, that did not prove to be the case.

o test our ability to match font requirements with font names we gathered font name information from several major vendors and application software. Names were extracted from fonts, from

published lists of fonts, and from tables of names provided by vendors.

While the results initially appear depressing – a common desktop environment can correctly render around 75% of a document collection – there are indications that with modest work the fonts required to faithfully render 92% of a collection can be readily identified. The problem of identification is unsolvable in an absolute sense – in any collection of documents there are likely to be required fonts that cannot be reasonably identified. There are simply too many fonts which have been in use and the data on font names available from font foundries or the fonts themselves too sparse to guarantee identification. Furthermore, some documents may include fonts with corrupt name information – we have seen examples of documents with indecipherable font names.

The remainder of this paper is organized as follows. We begin with a discussion of font formats (e.g. TrueType) and the available font identification information. By gathering information from fonts and font vendors, we have created a database containing several thousand popular fonts. We then describe the information available in Office documents, and our use of open source libraries to extract the names of fonts referenced by these documents. Finally, we describe experiments to match font names extracted from Microsoft Office documents with names in our database.

Background

A font consists of a set of glyphs indexed by *codepoints* (integers) within one or more *codepages* (a defined codepoint to a character mapping such as the latin alphabet) along with various geometric rendering information. Two fonts may be suitable substitutes if they contain similar glyphs for all codepoints. While this is frequently the case where the glyphs represent characters from common alphabets, there are many special case of symbolic characters (e.g. mathematical, scientific, or icons) where substitution of glyphs from another font destroys the underlying meaning of the document.

While there have been many font formats in use, the two most common formats for Windows platforms are PostScript and TrueType. There are still bitmapped fonts distributed with Windows for MS-DOS compatibility (FON files) which do appear to be used in some Office files. However, other than noting that they contain name strings that can be extracted, we do not discuss them further.

Although PostScript fonts appear to be in the decline, they were the dominant font format for at least a decade. There are several PostScript font formats (e.g. types 0, 1, and 2) but many of the differences relate to how glyphs are defined. For font identification, the key information provided in every PostScript font includes ASCII strings identifying the font version, family name, font name, and full name [21]. The full name shows the complete name of a typeface including style and character set information, and is typically used in font menus. The font name generally contains much of the same information as the full name, but in a compressed form limited to 29 characters. There are conventions for this compression (for example a rule that reduces the “words” to a string with 5,3 and 3 characters); however, it can be challenging to relate the font name strings with published lists of fonts [69].

TrueType was developed by Apple as a competitor to PostScript fonts and was subsequently adopted by Windows. Today, TrueType is the most common font format for Mac OS, the X Windows platform, and Microsoft Windows. The file format for TrueType is now covered by the OpenType specification (which can serve as a container for PostScript fonts). OpenType files are organized as a set of tables. The most important tables for our work are the naming tables (*name*) and the table (*OS/2*) containing Windows metrics including the Unicode and Windows code page ranges. The name table includes various strings keyed by platform, encoding, language, and name. Platforms include Windows and Macintosh. Encodings are platform specific – on the Macintosh these correspond to script manager codes (e.g. Roman, Japanese, etc.). On Windows, common encodings

include Unicode UCS-2 and UCS-4.

OpenType language IDs are platform-specific, and are used to indicate various language specific translations of the name strings. The types of names include copyright, font family, font subfamily, full font name, and PostScript names. The language used in font name strings within Office documents corresponds (where available) to the language for the Windows platform upon which the document was created; for example, “Arial Bold”, “Arial Negrita”, “Arial Vet”, and “Arial Gras” are the English, Spanish, Dutch, and French names for the same font. In gathering font data for this paper we found that only the most widely used fonts tend to have name strings in multiple languages and several large foundries provided only English names strings with the majority of their fonts.

Fonts in Windows and Word

In this paper we concentrate on the legacy Microsoft Word binary format. While we have performed no specific work with other formats (e.g. WordPerfect) there is good reason to believe that the conclusions will be similar because application programs such as Word and WordPerfect depend upon the underlying operating system API for access to and rendering of installed fonts. The font information embedded in a document is ultimately based upon the information available from the system API. Thus, we begin with a brief examination of the Win32 font functionality as described in the MSDN documentation [73]. A complete analysis examining the APIs of other operating systems (e.g. for the Macintosh) is beyond the scope of this paper.

The central data structure used by an application to exchange font information with the Windows operating system is the “logical font” or LOGFONT structure that is used to describe the most significant features of a font. Applications create LOGFONT structures to request that Windows find a matching font and Windows enumerates available fonts for applications by generating

LOGFONT structures. The structure provides information such as weight, orientation, and style (e.g. script, decorative, roman) as well as a (maximum) 32 character (Unicode or ASCII) name for the font. The metrics provide geometric information such as pitch and width, style information such as italic or underlined, and information about the range and type of the character set supported (e.g. ANSI, Symbol, Turkish). The type information can be used to distinguish “Raster fonts”, “Vector fonts”, “TrueType fonts”, and “Downloadable fonts”.

Word documents store their knowledge about fonts in a similar structure table of “font family names” (FFN), which include the name string (from the LOGFONT structure), a flag indicating whether the font is TrueType, the character set, the font weight and style, PANOSE number, and a 20-byte “font signature” [71]. The name string is either in UTF-16 or ASCII depending upon the version of Word creating a document. The font signature indicates the Unicode (16 bytes) and Windows code pages (8 bytes) for which the font contains glyphs. Similar data exist in the OS/2 metrics table for OpenType fonts and *might* be useful for verifying that a font file matches the font referenced by a Word document.

PANOSE numbers were developed in the 1980’s [3] as a mechanism for classifying fonts with the explicit goal of identifying good substitutes. While TrueType fonts include PANOSE numbers, and Word retains those numbers in its font tables, published research suggests that PANOSE has not been widely implemented correctly – many fonts have “default” values [48]. Furthermore, where PANOSE is correctly used the information is likely to be of little added value for font identification because these fonts tend to be those distributed by major vendors such as Microsoft for which accurate font name data has been recorded.

In our work, we rely upon the font name strings extracted from Office documents. While every Word document contains a single font name table, not all fonts listed in this table are used by the document. Furthermore, our experience suggests that as a document evolves, Word fails to

properly delete unused entries and empty or “garbage collect” name strings. Thus, extracting the set of valid name strings requires walking the document character by character.

Font Matching Experiments

In this Section we use two data sets totaling approximately 230,000 Word documents to evaluate the difficulty of identifying referenced fonts given a database of font information. The primary identification technique we utilize is based upon name matching; however, we also evaluate the utility of the other font metrics recorded in Word documents.

We had hoped to build a database of font information extracted from the fonts published by the major foundries; however, we found most foundries reluctant to provide the requested information and the cost of purchasing fonts not justified by this exploratory research.³ We resorted to combining information from a variety of sources including font files and font name information provided by several major foundries, extracted from published lists, and retrieved from foundry web pages. The font collections include fonts provided with various Mac OS X and MS Windows distributions, fonts distributed with applications such as Microsoft Office and WordPerfect, and fonts donated by Bitstream. The font data we collected are summarized in Table A.1.

To determine the fonts used in Word documents, we wrote a custom application based upon the open source library `libwv` which is the basis for file import in Abiword and other applications [55]. Through comparing key aspects of `libwv` with the published specification, we believe `libwv` is relatively correct – there are aspects of the specification that are far from clear, and specific notes point to items requiring further work. The `libwv` software provides a basic document processing function with application-specific callback functions. For our work, it was necessary only to

³We are grateful to Bitstream for providing a large collection of fonts, FontFont and URW for providing tables of font name information.

Foundry		
Adobe	Published Table	2374
Bitstream	TrueType Fonts	1556
FontFont	Foundry Supplied Table	11973
URW	Foundry Supplied Table	2358
Operating System		
Microsoft Windows + Office	Font Files	444
Mac OS X + Office	Font Files	322
Application		
Adobe PostScript 3 Fonts	Published List	103
Microsoft Applications	Published List	537
WordPerfect	TrueType Fonts	1080
Source	Data Type	Number of Fonts

Table A.1: Font Data

provide a function to track the fonts specified at the beginning of a text run and a function to process each character in a text run. For each font in the document name table, we record both the codepoints used and the number of characters referencing the font. Selection of the correct font for a given character is actually done incorrectly in `libwv` – the code does not implement the “font calculation” described in Appendix B of the Word 2007 Binary File Format document [71]. We corrected this error in our work. Our code generates reports including the active font names with the number of codepoints and characters from each font as well as the other font metrics recorded in the Word document.

Given the font information extracted from a collection of Word documents and a database of font names we can determine which extracted names *exactly* match names in the database. While there appear to be opportunities to apply various heuristics for inexact matching (for example longest matching prefixes), our examination of the names suggests there are many special cases, thus we studied exact matching as a baseline. We matched names against three distinct name sets – our complete collection, the names extracted from an installation of Windows XP and Office 2007, and the single name “Times New Roman” which is the most common font referenced in our data

sets.

Our metric for each font collection is the percentage of documents whose font requirements can be completely met (satisfied) by that collection. We compare these results with the percentage of “satisfied documents” given font collections of the N most referenced fonts for all values of N. The results for our glossary based collection (3910 fonts) are illustrated in Figure A.3 and those for the “.gov” (1920 fonts) documents are illustrated in Figure A.4. Although the total number of referenced fonts differs, the overall results are quite similar – roughly 31-39% satisfied by Times New Roman, 72-79% satisfied by XP and Office, and 90-94% satisfied by our more comprehensive collection. Notice that in both cases the top 100 fonts satisfy approximately 92% of the documents.

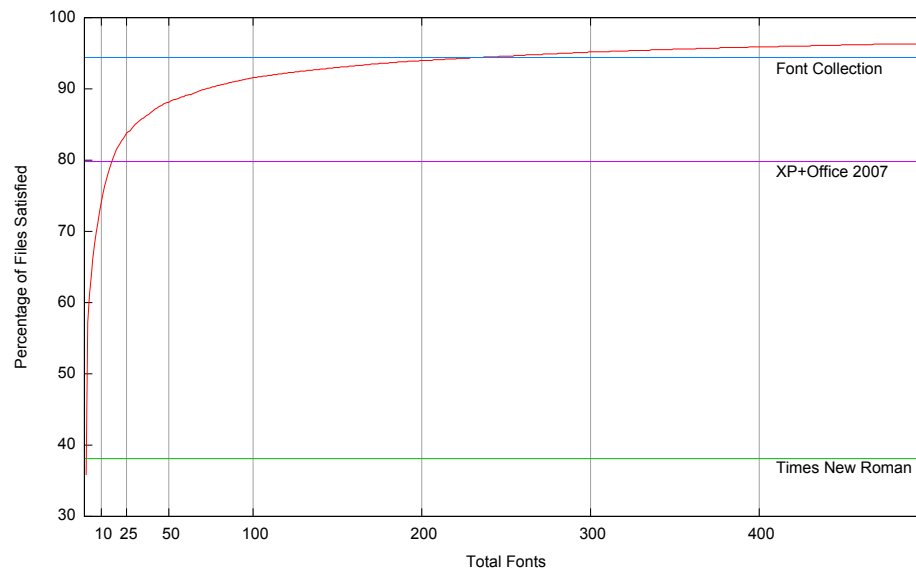


Figure A.3: Font Usage for Glossary Documents

As mentioned above, exact name matching is probably too pessimistic. For example, we noticed many variations on fonts including the word “Times”. Some of this variation is due to similar fonts published by different vendors, some is due to changes in name conventions from the early bitmapped fonts to PostScript fonts to TrueType, and some is due to significant differences. The top

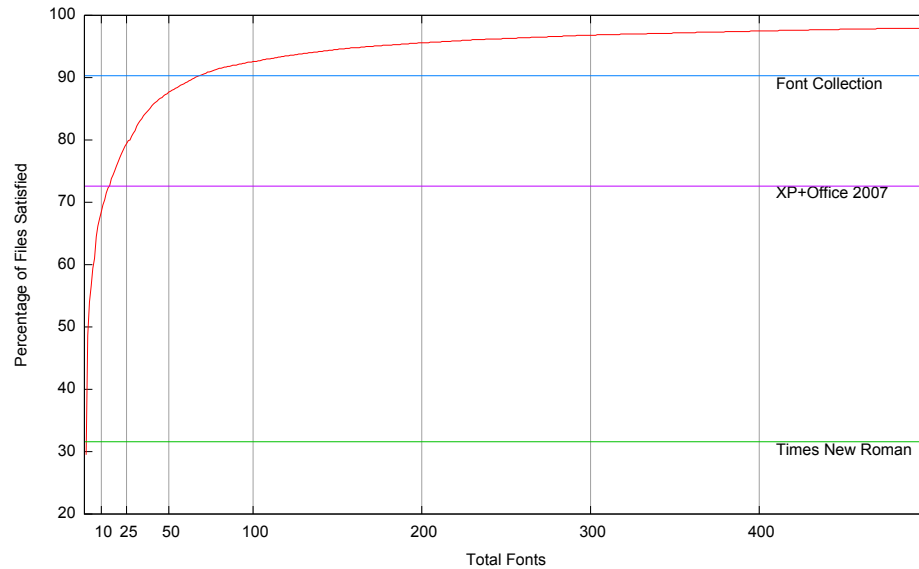


Figure A.4: Font Usage for Government Documents

10 “Times” fonts and their fraction of reference from the two document collections is illustrated in Table A.2. The complete list comprises 375 fonts and 49% of all font references. Note that the values in Table A.2 are calculated from the *total number of references* rather than the percentage of documents satisfied.

Even if all variations on Times, after suitable analysis, proved to be equivalent, the overall problem isn’t significantly simplified. The extremely long tail on font usage means achieving a 95% satisfaction level for a document collection is tractable, achieving 99% may not be feasible. Furthermore, our experience suggests that finding many of the identified fonts will be quite challenging.

Ultimately, preservation of documents will require selection of suitable font substitutes either because a particular font cannot be obtained or identified. Thus, we examined other data that might aid in characterizing suitable substitutes or in determining whether a required font is critical to preserving the information content of a document.

Times New Roman	42%
Times	3%
CG Times	1%
TimesNewRoman	0.5%
Times New Roman Bold	< 0.5%
TimesNewRoman,Bold	< 0.5%
Times New (W1)	< 0.5%
CG Times (W1)	< 0.5%
TimesNewRomanPSMT	< 0.5%
Times-Roman	< 0.5%
...	...
Total (375)	49%

Table A.2: Times Variations. Percentages calculated from number of times each font variation is encountered.

The primary additional font metrics available from Word documents include the font family (Roman, Swiss, etc.) and pitch, the font weight, Panose number, and Code sets (Unicode and Windows). The two font families that are likely to be safely substituted are Roman and Swiss (serif and san serif fonts respectively). It is less clear whether Modern or Script can be substituted, and Decorative generally cannot be substituted. As illustrated in Table A.3 and Table A.4, either the font family or pitch information has a “default” value for nearly 40% of all referenced fonts. The font family categories are described in [70].

Default	40%
Fixed Pitch	3%
Variable Pitch	57%

Table A.3: Font Pitch

Unicode range and Codepage information can be used to analyze font data at a finer granularity, providing a clearer picture of why a particular font may be used in a given document (e.g. in order to satisfy the need for particular glyphs) and potential guidance on selection of a suitable substitute. We intend to explore the use of such information in future research.

Default	Do not care or don't know	42%
Roman	Serif fonts with variable stroke width	28%
Swiss	San serif fonts with variable stroke width	21.5%
Modern	Constant stroke width	4.5%
Script	Handwriting	2%
Decorative	Novelty or other	2%

Table A.4: Font Family

As mentioned, many fonts are used for a single or few glyphs. In the case of the document collections studied, nearly 10% of all referenced fonts were never used for more than a single glyph of these. Unfortunately, even assuming all these fonts are known (by adding the names to our database) doesn't appreciably alter our "satisfaction" rate.

Font Substitutions

Locating missing fonts using name strings as a primary identifier is challenging. While a number of websites providing font search (one-at-a-time) based upon large databases, they are often incomplete, inaccurate, and cannot be automatically queried. We encountered a variety of problems with names containing non-ASCII characters, compressed names, variations of known names, and rare or specialized fonts. Processing compressed names in particular may be platform specific. For example, the Windows fonts named "Helv" and "Tms Rmn" in Windows 3.0 were renamed to "MS Sans Serif" and "MS Serif", respectively, in Windows 3.1. Modern iterations of the Microsoft operating system maintain a system-accessible map in the font substitution registry subkey to correctly map these names. However, on Macintosh platforms the proper substitutions remain unavailable.

Font substitution in Microsoft Windows is performed according to a “closest match” criteria calculated as a weighted sum from a vector of information corresponding to the LOGFONT structure. As the name suggests, this information does not correspond to a “physical” font (e.g. the data that define a font installed or loaded into the task environment) but rather a “logical” font composed of a sequence of properties requested as a result of a user action or application request during the loading of a document.

When an application is tasked with displaying text in a given font, it performs an API function call populated with values from the LOGFONT corresponding to the desired (logical) attributes. Windows realizes a best match to the desired font by searching the installed fonts to find one with values closest to this attribute set. As previously noted, these attributes include font family names, font weight, font width, and font slope. An obvious consequence of this is that the (absolute) best match may correspond to a font not installed on the system.

Font matching information is precalculated and cached from existing font collections according to a detailed algorithm designed to support efficient font matching on request. In addition to the attributes noted above, this process includes the generation of combined Family and Face names, extraction and resolution of various terms for style and weight (for example, “Bold Face” to be treated the same as “Bold”), and the extraction of canonical numerical representations.

Fonts are subsequently matched by location of an exact (or longest-substring match) FontFamily name, a matching face from the candidate face list (computed as a weighted attribute vector based on FontStretch, FontStyle, and FontWeight - prioritized in that order), and localization settings.

Microsoft provides detailed information on font matching in Windows under the Windows Presentation Foundation [72]. In some cases, a matched font may not contain the required glyphs (for example, a font with a Latin-only glyph-set requested to render glyphs in an unsupported codepoint range). The Microsoft Unicode-rendering service, Uniscribe, will automatically render

the unsupported script in the appropriate fallback font [52]. In addition to this “transparent” glyph substitution, fonts (in Windows 2000 and later) may also be *linked*, providing the ability to explicitly select add code points to a “base font” for which additional glyphs are desired.

In order to provide more accurate mapping from logical to physical fonts and account for improved rendering technologies, Microsoft introduced improved APIs in Windows XP (moving from GDI to GDI+) and Windows 7 (with the development of DirectWrite). As a result, the font substitution actions performed by different versions of Microsoft Office may be inconsistent. Additionally, the default font substitution algorithm implemented in GDI+ may be overridden within an application.

Because of this, it can be difficult to accurately replicate substitution actions performed by a proprietary application such as Microsoft Word. This can be demonstrated via a simple experiment using fonts provided with the operating system. If a LOGFONT structure is populated completely with information from a font loaded into the system font table using an API call such as `AddFontResourceEx()`, and that font is then unloaded, a subsequent call to the function `CreateFont()` under Windows XP (GDI+) will frequently result in mapping that differs from the substitution performed by Microsoft Office.

Furthering the problem, there is not enough information included in a legacy Word (.doc) file for any font used to completely fill the LOGFONT structure. Font information that *is* available, such as the PANOSE number or a substitution “hint”, cannot be passed directly to font mapping functions within GDI+. This is important because it exposes a significant preservation issue; even when document specifications are well documented (or fully open), the behavior of the application most commonly used to render those documents becomes the defacto standard for all rendering services. If the behavior of such an application depends on code or API functions that are not

generally exposed, the “openness” of the format is not necessarily a guarantee of preservation-friendliness.

David Levy notes that the severity of the risk posed by font substitution depends on a variety of factors, that “even in these simplest of cases, sensitivity to the circumstances of use is crucial to determining what is to be preserved - what counts as successful preservation” [59]. Font selection algorithms are complex, and the substitution actions performed may be opaque to anyone who is not a typographer or software developer. However, simple tools can assist the user in determining the degree to which a rendered document is well-formed (or exhibits information loss).

We extended the functionality of our `libwv` based tool with a small custom plug-in written in C# for Microsoft Word. This tool preprocesses the document as described in the Font Matching Experiments section, and presents the user with a dialog identifying the total number of document characters, a dump of the font name table along with the total number of glyphs in the font actually used in the document, and buttons to mark up (highlight) or search the document for specific font occurrences *irrespective* of whether they have been correctly rendered or subject to substitution.

In the example illustrated in Figure A.5, the unavailable Cyrillic font “Glasnost Light” has been substituted by a font which does not contain glyphs in the appropriate character range. This particular example uncovers another subtle problem with font identification. The publisher of the version of the Glasnost Light font used in this document had remapped an existing character range – in this case, standard Western ASCII characters for Cyrillic glyphs. We discovered this only after we had located what we believed to be the correct version of the font. The rendered document remained garbled (albeit this time with Cyrillic characters) due to the fact that we had selected a version of the font with the Cyrillic characters in the proper code page.

We believe that these tools can be incorporated into risk assessment workflows in a manner that is both low cost and highly reliable. Batch processing using our previously described tool can be

used to rapidly and automatically flag potential high-risk documents, which may then be presented to the user via the augmented Microsoft Word interface in order to determine the actual risk.

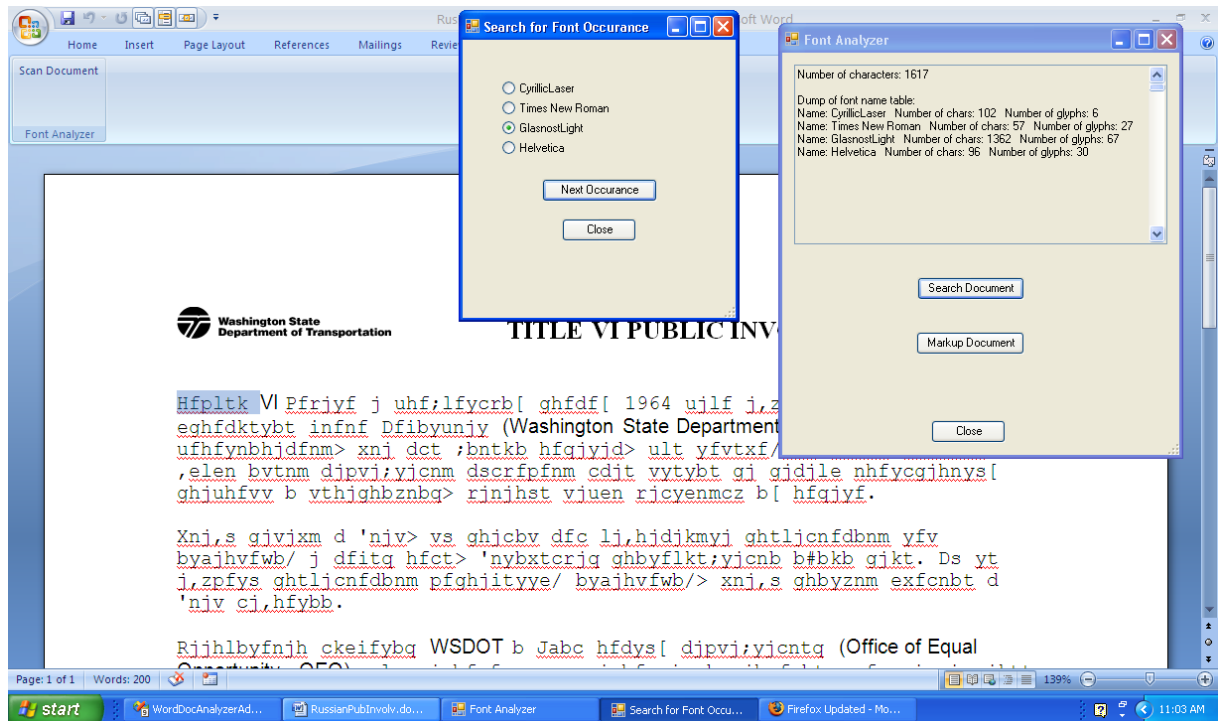


Figure A.5: Word plug-in identifying substituted “Glasnost Light” text.

Discussion

We have shown that the majority of digital documents obtained from a wide range of sources (up to 79%) can be rendered accurately using fonts appearing in modern desktop environments such as the combination of Microsoft Windows and Microsoft Office. With a small amount of additional work – using information drawn from font foundries, performing family name matches for legacy fonts or commercial fonts for which distribution has ceased, we can expect to increase this coverage to 92%.

This nevertheless leaves a large number of documents unaccounted for. Microsoft's own search engine indexes nearly 60 million documents currently available on the web. At this level of coverage, 1.8 million documents are guaranteed to be rendered inconsistently on a typical workstation. For many of these documents, the loss of information may be negligible. It is impossible, however, to quantify this without appropriate software tools to analyze the risk to a particular collection.

Even with access to the full documentation for a legacy proprietary format, replicating the behavior of the original environment used to render that document can be extremely difficult – or impossible – for tasks as seemingly basic as font selection and use. While publishers and institutional archives build support around work-flows optimized for “born archival” documents, the majority of the documents produced in the world today continue to be created using proprietary office software with font embedding disabled. With an open syntactic specification for the documents, the font identification and selection problem remains.

Automated tools to simplify the identification and location of missing fonts in document sets can significantly reduce the risk of information loss in an archive. As part of our ongoing research, we are developing tools to assist archivists in processing and analyzing font information from large collections of documents. Our current tool is capable of automatically preprocessing collections of Microsoft Word documents, identifying ranges of characters for which fonts are not available, and rendering those ranges separately from the remainder of the document for examination. In concert with this tool, we have developed a plug-in for Word which automatically locates and highlights ranges for which fonts or characters have been substituted. Future iterations of the tool may be adapted to use unicode ranges for language identification, or incorporate existing language identification software to assist in separation of human language and symbolic font uses.

Our research demonstrates the need for simple, effective tools to correctly identify font information and locate missing font data in order to facilitate lossless rendering. We show that effective

rendering of heterogeneous document collections can only occur when supported by a database of information drawn from multiple vendors; no existing identification technology provides universal or even adequate coverage. Proper archival handling of these digital objects should include tools to rapidly and selectively present to a human relevant document segments for quality assurance in order to mitigate risk during subsequent archival and access events.

A.2 Future Directions

As migration risk is often path specific, a critical source of information about features likely to result in information loss is operational logs generated by the various translation tools. Such issues are not restricted to legacy formats; both Microsoft and OpenOffice have documented extensive issues in translating between ‘open’ formats such as OfficeOpenXML and the Open Document Format.

One direction in future research will be to lower the human costs associated with risk mitigation in common archival tasks such as migration by providing interactive tools to facilitate evaluation of rendered documents both in their original environments and post-migration. Previous work by the author provides support for the efficacy of this approach even when using relatively simple tools such as plug-ins for modern office document suites. Such tools may be combined with statistical analysis of large document collections to identify and prioritize documents likely to be “high risk”.

Another method for addressing risks is to synthesize documents that exercise specific features of both legacy binary and modern formats in order to establish profiles for rapid and accurate identification of problem factors in real-world documents. Isolating these features will allow for testing to determine interaction between risk factors, and assist in building additional test cases for archival document management. be debugging rather than research.

Numerous existing publications provide taxonomic descriptions of digital object properties considered to be significant in a preservation context, or those which must be accounted for during transformation – migration from one format to another [5, 14, 29]. At least one formal information model for the treatment of document transforms can also be found [105]. A direction for future research will be to provide a scientific exploration of such properties – identifying features of digital objects that affect specific properties, developing processes to automatically identify and analyze those features, testing documents drawn from archival or online sources for the presence of these features, and automating risk-mitigation.

B

ISO Verification, Automated Digital Object Migration, and Emulation Wrappers

B.1 Overview

This appendix provides detailed information on the technologies and implementation strategies used to migrate digital materials found in the GPO collection to more modern renditions. This does not provide comprehensive coverage of the code developed as the format identification, migration, emulation, and web-access tools for the experiments described in the previous chapters. Many of the technologies used here – particularly the OpenOffice and VMware APIs – are being consistently updated. The code presented here focuses on some core technologies that demonstrate the low-cost, non-intrusive nature of the solutions previously examined.

B.2 ISO 9660 Truncation Test Tool

The following tool utilizes the CDIO package (with libiso9660) to perform a walk of the ISO-internal filesystem.

```
#define _FILE_OFFSET_BITS 64
#include <sys/types.h>
#include <cdio/cdio.h>
#include <cdio/iso9660.h>
#include <cdio/types.h>
```

```
#include <cdio/logging.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <openssl/md5.h>
#include <libgen.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <signal.h>

iso9660_t *p_iso;
iso9660_pvd_t p_pvd;
char *psz_fname;
const char *base;
uint8_t i_joliet_level;

int fildes;
int blocksize;

void catch_int(int sig_num)
{
    fprintf(stderr, "Segmentation fault --- %s\n", psz_fname);
    exit(1);
}

struct stat sb;

int file(iso9660_stat_t *p_statbuf, const char *path, const char *name)
{
    long int size = p_statbuf->size;
    int lsn = p_statbuf->lsn;

    if (sb.st_size < (long long) lsn * blocksize + size)
    {
        fprintf(stderr, "Fatal Error %s %s %s reading ISO 9660: File
            longer than image size = %d\n",
            psz_fname, path, name, size );
        // return 1;
        exit(1);
    }
    return 0;
}
```

```

int dir(const char *path, const char *dirname)
{
    char newpath[4096];
    CdioList_t *entlist;
    CdioListNode_t *entnode;

    if ((strcmp(dirname, ".") == 0) || (strcmp(dirname, "..") == 0))
        return;

    sprintf(newpath, "%s%s/", path, dirname);
    entlist = iso9660_ifs_readdir (p_iso, newpath);

    /* Iterate over the list of nodes that iso9660_ifs_readdir gives */

    if (entlist) {
        _CDIO_LIST_FOREACH (entnode, entlist)
        {
            char filename[4096];
            iso9660_stat_t *p_statbuf =
                (iso9660_stat_t *) _cdio_list_node_data (entnode);
            iso9660_name_translate_ext(p_statbuf->filename, filename,
                                      i_joliet_level);
            if (p_statbuf->type == _STAT_FILE)
            {
                if (file(p_statbuf, newpath, filename))
                {
                    _cdio_list_free (entlist, true);
                    return(1);
                }
            }
            else
            {
                if (strcmp(p_statbuf->filename, "") && !file(p_statbuf,
                                                              newpath,
                                                              filename) &&
                    dir(newpath,
                       p_statbuf->filename)
                )
                {
                    _cdio_list_free (entlist, true);
                    return(1);
                }
                // check directory file length too !
            }
        }
        _cdio_list_free (entlist, true);
    }
}

```

```
    return(0);
}
else
{
    fprintf(stderr, "Fatal_Error_%s_Couldn't_open_%s\n", psz_fname,
            newpath);
    return(1);
}
}

void loghandler(cdio_log_level_t level, const char message[])
{
    if ((level == CDIO_LOG_ERROR) || (level == CDIO_LOG_FATAL))
    {
        fprintf(stderr, "Critical_Error_%s_%s\n", psz_fname, message);
    }
}

int main(int argc, const char *argv[])
{
    signal(SIGSEGV, catch_int);
    cdio_log_set_handler(loghandler);

    if (argc < 2)
        exit(0);
    psz_fname = strdup(argv[1]);
    p_iso = iso9660_open_ext (psz_fname, ISO_EXTENSION_NONE);

    if (NULL == p_iso) {
        fprintf(stderr, "Fatal_Error_%s_could_not_find_an_ISO_9660_image\n",
                psz_fname);
        iso9660_close(p_iso);
        exit(1);
    }

    fildes = open64(psz_fname, ORDONLY);
    if (fildes < 0)
    {
        perror("problem_with_open");
    }
    fstat(fildes, &sb);
    i_joliet_level = iso9660_ifs_get_joliet_level(p_iso);
    base = basename(psz_fname);

    if (iso9660_ifs_read_pvd (p_iso, &p_pvd))
        blocksize = iso9660_get_pvd_block_size(&p_pvd);
    else
```

```
    {
        fprintf(stderr, "Problem reading pvd\n");
        exit(1);
    }

    dir("", "");
    iso9660_close(p_iso);
    exit(0);
}
```

B.3 Migration Automation

The migration automation code used to provide document migration services in the research described in Chapter 5 is based primarily on open sample code provided to the OpenOffice developer community in the form of a simple Python daemon designed to ingest and interpret commonly used but proprietary document formats such as pre-OfficeOpenXML Microsoft Word and Excel documents. Additional modifications were made using source from the open ERP5 resource management system, which includes a multi-dispatch version of the conversion daemon to conduct simultaneous jobs.

Because both the Python language, the OpenOffice APIs, and the tools and documentation to create OpenOffice extensions (UNO components) continue to evolve, the code is not included in this document, but can be supplied on request. The ERP5 code sources can be found via the subversion repository at <https://svn.erp5.org/repos/public/erp5/trunk/utils/oood/>. An LGPL alpha implementation of a simpler migration daemon can be found at <http://udk.openoffice.org/python/oood/>.

B.4 Emulation Automation

The complete C# code used to guide the user through creation of customized emulation environments is not included here. The following code provides simple automation access for identifying, powering on, copying files to, and switching contexts for the virtual machine.

```
using System.Runtime.InteropServices;
using VixCOM;

namespace AFSBrowser {
    public class VixWrapper {
        VixCOM.IVixLib vixLib = null;

        ulong m_vixError;
        VixCOM.IHost m_hostHandle = null;
        VixCOM.IVM m_vmHandle = null;

        public ulong GetError() {
            return m_vixError;
        }

        public VixWrapper() {
            try {
                vixLib = new VixCOM.VixLibClass();
            }
            catch (COMException comExc) {
                System.Diagnostics.Trace.WriteLine(comExc.Message + "\n");
                throw;
            }
        }

        /// <summary>
        /// Creates a host handle
        /// </summary>
        /// <returns>true if succeeded, otherwise false </returns>
        public bool Connect(string hostName, string userName, string
            password, int serviceProvider) {
            int hostType = -1;
            // Edited by Mitchell Lutz
            if (serviceProvider == 1) {
                hostType = VixCOM.Constants.
                    VIX.SERVICEPROVIDER.VMWARE.WORKSTATION;
            }
        }
    }
}
```

```

else if(serviceProvider == 2) {
    hostType = VixCOM.Constants.
        VIX.SERVICEPROVIDER.VMWARE.SERVER;
}
//int hostType = string.IsNullOrEmpty(hostName) ?
//    VixCOM.Constants.
//        VIX.SERVICEPROVIDER.VMWARE.WORKSTATION :
//    VixCOM.Constants.VIX.SERVICEPROVIDER.VMWARE.SERVER;

int vixVersion = VixCOM.Constants.VIX.API.VERSION;

vixVersion = 1; // Bugfix: http://communities.vmware.com/
    message/649925#649925

VixCOM.IJob jobHandle = vixLib.Connect(vixVersion,
    hostType, hostName, 0, userName, password, 0, null, null
    );

int[] propertyIds = new int[1] { VixCOM.Constants.
    VIX.PROPERTY_JOB.RESULT_HANDLE };
object results = new object();

m_vixError = jobHandle.Wait(propertyIds, ref results);

if(m_vixError == VixCOM.Constants.VIX.OK) {
    object[] objectArray = (object[]) results;
    m_hostHandle = (VixCOM.IHost)objectArray[0];
    return true;
}

return false;
}

/// <summary>
/// Opens the virtual machine specified in vmxFilePath
/// </summary>
/// <param name= vmxFilePath >The virtual machine vmx file
    to open</param>
/// <returns>true if succeeded, otherwise false</returns>
public bool Open(string vmxFilePath) {
    IJob jobHandle = m_hostHandle.OpenVM(vmxFilePath, null);

    int[] propertyIds = new int[1] { VixCOM.Constants.
        VIX.PROPERTY_JOB.RESULT_HANDLE };
    object results = new object();

    m_vixError = jobHandle.Wait(propertyIds, ref results);

```



```

        if(m_vixError == VixCOM.Constants.VIX_OK) {
            object[] objectArray = (object[]) results;
            m_vmHandle = (VixCOM.IVM) objectArray[0];
            return true;
        }

        return false;
    }

    /// <summary>
    /// Power on the virtual machine
    /// </summary>
    /// <returns>true if succeeded, otherwise false </returns>
    public bool PowerOn() {
        IJob jobHandle = m_vmHandle.PowerOn(VixCOM.Constants.
            VIX.VMPOWEROP.LAUNCH.GUI,
            null, null);
        m_vixError = jobHandle.WaitWithoutResults();

        if(m_vixError == VixCOM.Constants.VIX_OK) {
            //
            // Wait until guest is completely booted.
            //
            jobHandle = m_vmHandle.WaitForToolsInGuest(300, null);

            m_vixError = jobHandle.WaitWithoutResults();
        }

        return (m_vixError == VixCOM.Constants.VIX_OK);
    }

    /// <summary>
    /// Starts a snapshot of a virtual machine
    /// </summary>
    /// <param name= snapshot_name >The name of the snapshot to
    start </param>
    /// <returns>true if succeeded, otherwise false </returns>
    public bool RevertToLastSnapshot() {
        ISnapshot snapshot = null;

        m_vixError = m_vmHandle.GetRootSnapshot(0, out snapshot);

        if(m_vixError == VixCOM.Constants.VIX_OK) {
            IJob jobHandle = m_vmHandle.RevertToSnapshot(snapshot,
                0, null, null);

            m_vixError = jobHandle.WaitWithoutResults();
        }
    }

```

```

    }
    return (m_vixError == VixCOM.Constants.VIX_OK);
}

/// <summary>
/// Login to the virtual machine
/// </summary>
/// <returns>true if succeeded, otherwise false </returns>
public bool Login(string username, string password) {
    IJob jobHandle = m_vmHandle.LoginInGuest(username, password,
        0, null);
    m_vixError = jobHandle.WaitWithoutResults();

    return (m_vixError == VixCOM.Constants.VIX_OK);
}

/// <summary>
/// Creates the directory in the Virtual Machine
/// </summary>
/// <param name= pathName ></param>
/// <returns></returns>
public bool CreateDirectoryInVm(string pathName) {
    IJob jobHandle = m_vmHandle.CreateDirectoryInGuest(pathName,
        null, null);
    m_vixError = jobHandle.WaitWithoutResults();

    return (m_vixError == VixCOM.Constants.VIX_OK);
}

/// <summary>
/// Copies a file from the host machine to the virtual machine
/// </summary>
/// <param name= sourceFile >The source file on the host
machine</param>
/// <param name= destinationFile >The destination on the VM
</param>
/// <returns>true if succeeded, otherwise false </returns>
public bool CopyFileToVm(string sourceFile, string
destinationFile) {
    //
    // Copy files from host to guest
    //
    IJob jobHandle = m_vmHandle.CopyFileFromHostToGuest(
        sourceFile, destinationFile,
        0, null, null);
    m_vixError = jobHandle.WaitWithoutResults();

    return (m_vixError == VixCOM.Constants.VIX_OK);
}

```

```

}

/// <summary>
/// Copies a file from the virtual machine to the host machine
/// </summary>
/// <param name= sourceFile >The source file on the virtual
machine</param>
/// <param name= destinationFile >The destination on the
host machine</param>
/// <returns>true if succeeded, otherwise false</returns>
public bool CopyFileFromVm(string sourceFile, string
destinationFile) {
    //
    // Copy files from host to guest
    //
    IJob jobHandle = m_vmHandle.CopyFileFromGuestToHost(
        sourceFile, destinationFile,
        0, null, null);
    m_vixError = jobHandle.WaitWithoutResults();

    return (m_vixError == VixCOM.Constants.VIX_OK);
}

/// <summary>
/// Runs a program on the virtual machine
/// </summary>
/// <param name= exePath >The path of the program on the
virtual machine</param>
/// <param name= parameters >The parameters to pass to the
executable</param>
/// <param name= resultCode >The result code returned from
the program that ran on the VM</param>
/// <returns>true if succeeded, otherwise false</returns>
public bool RunProgram(string exePath, string parameters, out
int resultCode) {
    resultCode = -1;

    IJob jobHandle = m_vmHandle.RunProgramInGuest(exePath,
parameters, VixCOM.Constants.
VIX.RUNPROGRAMACTIVATE.WINDOW, null, null); //
clientData

    int[] propertyIds = new int[1] { VixCOM.Constants.
VIX.PROPERTY_JOB_RESULT_GUEST_PROGRAM_EXIT_CODE };
    object results = new object();
    m_vixError = jobHandle.Wait(propertyIds, ref results);

    if (m_vixError == VixCOM.Constants.VIX_OK) {

```

```
        object[] objectArray = (object[]) results;
        resultCode = (int) objectArray[0];
        return true;
    }

    return false;
}

/// <summary>
/// Power off the virtual machine
/// </summary>
/// <returns>true if succeeded, otherwise false </returns>
public bool PowerOff() {
    IJob jobHandle = m_vmHandle.PowerOff(VixCOM.Constants.
        VIX.VMPOWEROP_NORMAL, null);
    m_vixError = jobHandle.WaitWithoutResults();

    return (m_vixError == VixCOM.Constants.VIX.OK);
}
}
```

Bibliography

- [1] W.Y. Arms, D. Hillmann, C. Lagoze, D. Krafft, R. Marisa, J. Saylor, C. Terizzi, H. Van de Sompel, T. Gill, P. Miller, et al. A Spectrum of Interoperability: The Site for Science Prototype for the NSDL; Re-Inventing the Wheel? Standards, Interoperability and Digital Cultural Content; Preservation Risk Management for Web Resources: Virtual Remote Control in Cornell. *D-Lib Magazine*, 8:n1, 2002.
- [2] David Bainbridge, Ian Witten, Stefan Boddie, and John Thompson. Stress-testing general purpose digital library software. *Research and Advanced Technology for Digital Libraries*, 5714:203–214, 2009.
- [3] Benjamin Bauermeister. *A Manual of Comparative Typography*. Van Nostrand Reinhold, 1987.
- [4] C. Becker, A. Rauber, V. Heydegger, J. Schnasse, and M. Thaller. Systematic characterisation of objects in digital preservation: The extensible characterisation languages. *Journal of Universal Computer Science*, 14(18):2936–2952, 2008.
- [5] Christoph Becker, Andreas Rauber, Volker Heydegger, Jan Schnasse, and Manfred Thaller. A generic XML language for characterising objects to support digital preservation. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 402–406, New York, NY, USA, 2008. ACM.

- [6] Margret Branschofsky and Daniel Chudnov. Dspace: durable digital documents. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 372–372, New York, NY, USA, 2002. ACM.
- [7] Adiran Brown. Digital Preservation Technical Paper 1: Automatic Format Identification Using PRONOM and DROID. Technical report, National Archives, UK, March 2006.
- [8] Geoffrey Brown and Kam Woods. Born broken: Fonts in information loss in legacy digital documents. In *Sixth International Conference on Preservation of Digital Objects(iPRES 2009)*, October 2009.
- [9] Fred Byers. Care and Handling of CDs and DVDs. A Guide for Librarians and Archivists., 2003. <http://ftp.fatv.org/downloads/CDandDVDCareandHandlingGuide.pdf>.
- [10] P. Caplan. The Florida Digital Archive and DAITSS: a working preservation repository based on format migration. *International Journal on Digital Libraries*, 6(4):305–311, 2007.
- [11] Priscilla Caplan. Understanding PREMIS. Technical report, Library of Congress, December 2009.
- [12] Regional depository CD/DVD database. http://www.uky.edu/Libraries/CDROM_2004_inventory.xls. Accessed August, 2007.
- [13] The Common Data Format (CDF). <http://cdf.gsfc.nasa.gov/>. Accessed August 2007.
- [14] Carol C.H. Chou. Format Identification, Validation, Characterization, and Transformation in DAITSS. In *IS&T Archiving 2007*, 2007.
- [15] Lars Clausen. Opening Schrodingers Library: Semi-automatic QA Reduces Uncertainty in Object Transformation. In Laszlo Kovacs, Norbert Fuhr, and Carlo Meghini, editors, *Research*

- and Advanced Technology for Digital Libraries*, volume 4675 of *Lecture Notes in Computer Science*, pages 186–197. Springer Berlin / Heidelberg, 2007.
- [16] Lars R. Clausen. Opening Schrödingers Library: Semi-automatic QA Reduces Uncertainty in Object Transformation. In *Research and Advanced Technology for Digital Libraries*, volume 4675, pages 186–197. Springer Berlin/Heidelberg, 2007.
- [17] John Garrett (co chair) and Donald Walters (chair). Preserving digital information: Report on the task force on archiving of digital information, 1996. The Commission on Preservation and Access, The Research Libraries Group.
- [18] Codabar Symbology. <http://www.barcodeisland.com/codabar.phtml>. Accessed August, 2010.
- [19] CQL: Contextual Query Language. <http://www.loc.gov/standards/sru/specs/cql.html>. Accessed August 2007.
- [20] Joseph Curtis. AONS System Documentation Revision 1692006-09-29. Technical report, Australian Partnership for Sustainable Repositories, September 2006.
- [21] Technical Notes on Fonts, 2009. <http://www.adobe.com/devnet/font>.
- [22] Django: The Web framework for perfectionists with deadlines. <http://www.djangoproject.com>. Accessed August 2007.
- [23] Design Criteria Standard for Electronic Record Management Software Applications (DoD 5015.2-STD). <http://www.dtic.mil/whs/directives/corres/html/50152std.htm>. Accessed September, 2006.
- [24] Jhove Integration with DSpace. <http://wiki.dspace.org/index.php/JhoveIntegration>. Accessed November, 2006.

- [25] Richard Entlich and Ellie Buckley. Digging Up Bits of the Past: Hands-on With Obsolescence. *RLG DigiNews*, 10(5):13, 2006.
- [26] Deprecated Features for Excel 2007. <http://blogs.msdn.com/excel/archive/2006/08/24/718786.aspx>. Accessed November, 2006.
- [27] About the FDLP. http://www.access.gpo.gov/su_docs/fdlp/about.html. Accessed November, 2006.
- [28] CIC Floppy Disk Project. <http://www.indiana.edu/~libgpd/mforms/floppy/floppy.html>. Accessed September, 2006.
- [29] Miguel Ferreira. Automatic evaluation of migration quality in distributed networks of converters. *TCDL Bulletin*, 2(2), 2006.
- [30] FFmpeg: Cross-platform solution to stream audio and video. <http://ffmpeg.mplayerhq.hu>. Accessed August 2007.
- [31] Fine Free File Command. <http://www.darwinsys.com/file>. Accessed September, 2006.
- [32] Consultative Committee for Space Data Systems. Reference model for an open archival information system (OAIS). Technical Report CCSDS 650.0-B-1, CCSDS, January 2002.
- [33] Fred: A Format Registry Demonstration. <http://tom.library.upenn.edu/fred/>. Accessed November, 2006.
- [34] Gretchen Gano and Julie Linden. Government information in legacy formats: Scaling a pilot project to enable long-term access. *D-Lib Magazine*, 13(7):5, 2007.
- [35] Global Digital Format Registry. <http://hul.harvard.edu/gdfr/>. Accessed November, 2006.

- [36] Henry Gladney. *Preserving Digital Information*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [37] Kevin Glick, Eliot Wilczek, and Robert Dockins. Fedora and the Preservation of University Records Project. *RLG DigiNews*, 10(5):7, 2006.
- [38] Welcome to Gnumeric! <http://www.gnome.org/projects/gnumeric/>. Accessed December, 2007.
- [39] Depository Library Public Service Guidelines For Government Information in Electronic Formats. http://www.access.gpo.gov/su_docs/fdlp/mgt/pseguide.html. Accessed November, 2006.
- [40] Stewart Granger. Emulation as a Digital Preservation Strategy. *D-Lib Magazine*, 6(10), October 2000.
- [41] Siegfried Hackel. ArchiSafe: A Legally Secure and Scalable Long-Term Record-Keeping Strategy. *Proceedings of the DLM-FORUM 2005 - Preservation and Digital Archives*, 2005.
- [42] Maragaret Hedstrom and Christopher Lee. Significant properties of digital objects: Definitions, applications, and implications. In *Proceedings of the DLM-Forum*, pages 218–223, 2002.
- [43] Margaret Hedstrom, editor. *It's About Time: Research Challenges in Digital Archiving. Final Report Workshop on Resaerch Challenges in Digital Archiving and Long-Term Preservation*. National Science Foundation, August 2003.
- [44] Margaret Hedstrom and Clifford Lampe. Emulation vs. Migration: Do Users Care? *RLG DigiNews*, 5(6):9, 2002.
- [45] Gail Hodge and Nikkia Anderson. Formats for digital preservation: A review of alternatives and issues. *Inf. Serv. Use*, 27(1-2):45–63, 2007.

- [46] J. R. Van Der Hoeven, R. J. Van Diessen, and K. Van Der Meer. Development of a universal virtual computer (uvc) for long-term preservation of digital objects. *Journal of Information Science*, 31(3):196–208, 2005.
- [47] M. Ide, D. MacCarn, T. Shepard, and L. Weisse. Understanding the Preservation Challenge of Digital Television. In *Building a National Strategy for Preservation: Issues in Digital Media Archiving*. Council on Library and Information Resources and the Library of Congress, April 2002.
- [48] Jeremy Impson. Evaluating the IBM and HP/PANOSE font classification systems. *Online Information Review*, 29(5):14, 2005.
- [49] Federal Documents Depository Program. <http://www.statelib.lib.in.us/WWW/isl/rl/feddoc.html>. Accessed November, 2006.
- [50] Perla Innocenti, Seamus Ross, Elena Maceviciute, Tom Wilson, Jens Ludwig, and Wolfgang Pempe. Assessing digital preservation frameworks: the approach of the shaman project. In *MEDES '09: Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, pages 412–416, New York, NY, USA, 2009. ACM.
- [51] JHOVE - JSTOR/Harvard object validation environment. <http://hul.harvard.edu/jhove/>. Accessed September, 2006.
- [52] Michael Kaplan. Font substitution and linking #1, 2009. <http://blogs.msdn.com/-michkap/archive/2005/03/20/-399322.aspx>.
- [53] T. J. Killian. Processes as files. In *USENIX Association. Proceedings of the Summer 1984 USENIX Conference*, pages 203–207, Berkeley, CA, USA, 1984. USENIX.

- [54] kopal library for retrieval and ingest (koLibRI). http://kopal.langzeitarchivierung.de/index_koLibRI.php.en. Accessed August 2007.
- [55] Dom Lachowicz. wvWare, library for converting Word documents, 2009. <http://wvware.sourceforge.net/>.
- [56] Carl Lagoze, Sandy Payette, Edwin Shin, and Chris Wilper. Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6(2):124–138, April 2006.
- [57] Gregory W. Lawrence, William R. Kehoe, Oya Y. Rieger, Willam H. Walters, and Anne R. Kenney. Risk management of digital information: A file format investigation. Technical report, Council on Library and Information Resources, Washington D.C., June 2000.
- [58] Cal Lee. Never optimize: Building and managing a robust cyberinfrastructure. Technical report, University of Michigan, Ann Arbor, MI, June 2006.
- [59] David M. Levy. Heroic measures: reflections on the possibility and purpose of digital preservation. In *DL '98: Proceedings of the third ACM conference on Digital libraries*, pages 152–161, New York, NY, USA, 1998. ACM.
- [60] Sustainability of Digital Formats Planning for Library of Congress Collections. <http://www.digitalpreservation.gov/formats/index.shtml>. Accessed November, 2006.
- [61] Raymond A. Lorie. Long term preservation of digital information. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 346–352, New York, NY, USA, 2001. ACM Press.
- [62] Petros Maniatis, David S. H. Rosenthal, Mema Roussopoulos, and Mary Baker. Lockss: A

- peer-to-peer digital preservation system. *ACM Transactions on Computer Systems*, 23:2005, 2003.
- [63] MARC 21 XML schema. <http://www.loc.gov/standards/marcxml/>. Accessed November, 2006.
- [64] B. D. McCullough. Fixing Statistical Errors in Spreadsheet Software: The Cases of Gnumeric and Excel, 2004. www.csdassn.org/software_reports.cfm. Accessed November 2008.
- [65] Jerome P. McDonough. Aligning mets with the oai-ore data model. In *JCDL '09: Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 323–330, New York, NY, USA, 2009. ACM.
- [66] P. McDonough. Mets: standardized encoding for digital library objects. *Int. J. Digit. Libr.*, 6(2):148–158, 2006.
- [67] Phil Mellor, Paul Wheatley, and Derek M. Sergeant. Migration on Request, a Practical Technique for Preservation. In *ECDL '02: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, pages 516–526, London, UK, 2002. Springer-Verlag.
- [68] Pil Mellor. CaMiLEON: emulation and BBC doomsday. *RLG DigiNews*, 7(2), 2003.
- [69] List of fonts supplied with Microsoft products, 2009. <http://www.microsoft.com/typography/fonts/product.aspx?PID=1>.
- [70] Microsoft. LOGFONT, 2009. <http://msdn.microsoft.com/en-us/library/ms901140.aspx>.
- [71] Microsoft Word 2007 Binary Format Specification, 2009. [http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD.../Word97-2007BinaryFileFormat\(doc\)Specification.pdf](http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD.../Word97-2007BinaryFileFormat(doc)Specification.pdf).
- [72] WPF font selection model, 2009. blogs.msdn.com/text/attachment/2249036.ashx.

- [73] DirectWrite API Reference, 2009. [http://msdn.microsoft.com/en-us/library/dd368038\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/dd368038(VS.85).aspx).
- [74] Considerations for server-side Automation of Office. <http://support.microsoft.com/kb/257757>. Accessed August 2007.
- [75] MODS metadata object description schema. <http://www.loc.gov/standards/mods/>. Accessed November 17, 2006.
- [76] Reagan Moore. Building Preservation Environments with Data Grid Technology. <http://www.archives.gov/era/pdf/2006-saa-moore.pdf>. Accessed November, 2006.
- [77] 2005 Minimum Technical Requirements for Public Access Workstations in Federal Depository Libraries. http://www.access.gpo.gov/su_docs/fdlp/computers/mtr.html. Accessed November, 2006.
- [78] Endorsement of DoD electronics records management application (RMA) design criteria standard, version 2. NARA Bulletin 2003-03
<http://www.archives.gov/records-mgmt/bulletins/2003/2003-03.html>. Accessed September, 2006.
- [79] The Open Archives Initiative Protocol for Metadata Harvesting. <http://www.openarchives.org/OAI/openarchivesprotocol.html>. Accessed December, 2006.
- [80] Library of Congress. Search/Retrieval via URL, 2009. <http://www.loc.gov/standards/sru/>.
- [81] Erik Oltmans and Nanda Kol. A Comparison Between Migration and Emulation in Terms of Costs. *RLG DigiNews*, 9(2):10, 2005.
- [82] The OpenAFS Network Filesystem. <http://www.openafs.org/>. Accessed August, 2010.

- [83] OpenOffice: Free Office Suite. <http://www.openoffice.org>. Accessed on September, 2006.
- [84] David Pearson. AONS II: Continuing the Trend Towards Preservation Software 'Nirvana'. In *New Technology of Library and Informations Service iPRES2007 Special Issue*, pages 42–49, 2008. <http://www.infotech.ac.cn/qikan/public/pdfdown.asp?xiazailx=29&houzhui=.pdf>.
- [85] David Pearson and Colin Webb. Defining File Format Obsolescence: A Risky Journey. *International Journal of Digital Curation*, 3(1), 2008.
- [86] J. Pendry and M. McKusick. Union mounts in 4.4BSD-Lite. In *Proceedings of the New Orleans Usenix Conference*, January 1995.
- [87] Rob Pike, Dave Presotto, Sean Dorward, Bob Flandrena, Ken Thompson, Howard Trickey, and Phil Winterbottom. Plan 9 from Bell Labs. *Computing Systems*, 8(3):221–254, Summer 1995.
- [88] PLANETS: Integrated Services for Digital Preservation. *International Journal of Digital Curation*, 2008. <http://www.ijdc.net/index.php/ijdc/article/view/45>. Accessed December 2008.
- [89] PREMIS (PREservation Metadata: Implementation Strategies) working group. <http://www.oclc.org/research/projects/pmwg/>. Accessed November, 2006.
- [90] The technical registry PRONOM. <http://www.nationalarchives.gov.uk/pronom/>. Accessed November, 2006.
- [91] Carl Rauch and Andreas Rauber. Preserving digital media: Towards a preservation solution

- evaluation metric. In *In Proceedings of the 7th International Conference on Asian Digital Libraries, ICADL 2004*, pages 203–212. Springer, 2004.
- [92] K. Rechert, D. von Suchodoletz, R. Welt, M. van den Dobbsteven, B. Roberts, J van der Hoeven, and J. Schroder. Novel workflows for abstract handling of complex interaction processes in digital preservation. In *iPRES 2009: Proceedings of the Sixth International Conference on the Preservation of Digital Objects*, October 2009.
- [93] Thomas Reichherzer and Geoffrey Brown. Quantifying software requirements for supporting archived office documents using emulation. In *JCDL '06: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 86–94, New York, NY, USA, 2006. ACM.
- [94] David S. Rosenthal, Thomas Lipkis, Thomas S. Robertson, and Seth Morabito. Transparent Format Migration of Preserved Web Content. *D-Lib Magazine*, 11(1), January 2005.
- [95] Jeff Rothenberg. Ensuring the longevity of digital information. *Scientific American*, 272(1):42–47, January 1995.
- [96] Jeff Rothenberg. *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation*. Council on Library & Information Resources, 1999.
- [97] Jeff Rothenberg. An experiment in using emulation to preserve digital publications. Technical report, Koninklijke Bibliotheek, July 2000.
- [98] Jeff Rothenberg. Using emulation to preserve digital documents. Technical report, Koninklijke Bibliotheek, July 2000.
- [99] Chris Rusbridge. Excuse Me... Some Digital Preservation Fallacies? *Ariadne*, 46:9, 2006.
- [100] F. A. Salomon and D. A. Tafuri. Emulation - a useful tool in the development of computer systems. In *Proceedings of the fifteenth annual simulation symposium*, pages 55–71, 1982.

- [101] Seeing Double: Emulation Theory and Practice. <http://www.variablemedia.net/e/seeingdouble/home.html>. Accessed November, 2006.
- [102] Andreas Stanescu. Assessing the Durability of Formats in a Digital Preservation Environment. *OCLC Systems & Services: International Digital Library Perspectives*, 21(1):61–81, 2005.
- [103] Stephan Strodl, Christoph Becker, Robert Neumayer, and Andreas Rauber. How to choose a digital preservation strategy: evaluating a preservation planning procedure. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 29–38, New York, NY, USA, 2007. ACM.
- [104] Dave Thompson. A Pragmatic Approach to Preferred File Formats for Acquisition. *Ariadne*, 63:8, 2010.
- [105] Thomas Triebsees and Uwe M. Borghoff. A Theory for Model-Based Transformation Applied to Computer-Supported Preservation in Digital Archives. *Engineering of Computer-Based Systems, IEEE International Conference on the*, 0:359–370, 2007.
- [106] Survey and Assessment of Sources of Information on File Formats and Software Documentation. Technical report, The Representation and Rendering Project, University of Leeds, 2007.
- [107] Jeffrey van der Hoeven. Dioscuri: emulator for digital preservation. *D-Lib Magazine*, 13(11/12), November/December 2007.
- [108] Jeffrey van der Hoeven and Hilde van Wijngaarden. Modular emulation as a long-term preservation strategy for digital objects. *Proceedings of the International Web Archiving Workshop*, 2005. <http://www.iwaw.net/05/papers/iwaw05-hoeven.pdf>. Accessed December 2008.
- [109] Hilde van Wijngaarden and Erik Oltmans. Digital Preservation and Permanent Access:

- The UVC for Images. *Proceedings of the Imaging Science & Technology Archiving Conference, 2004*. http://www.kb.nl/hrd/dd/dd_links_en_publicaties/publicaties/uvc-ist.pdf. Accessed December 2009.
- [110] D von Suchodoletz and J. van der Hoeven. Emulation: From digital artifact to remotely rendered environments. In *iPRES 2008: Proceedings of the Fifth International Conference on Preservation of Digital Objects*, pages 93–98, 2008.
- [111] Paul Wheatley. Migration – a CAMiLEON discussion paper. *Ariadne*, (29), September 2001.
- [112] Identifying Fonts for Digital Document Preservation. <http://docs.google.com/gview?a=v&attid=0.1&thid=11faf4db9a21c4b3&mt=application%2Fpdf>. Accessed March 2009.
- [113] Ian Witten, David Bainbridge, and Stefan Boddie. Greenstone Open-Source Digital Library Software. *D-Lib Magazine*, 7(10):8, 2001.
- [114] K. Woods and G. Brown. Migration Performance for Legacy Data Access. *International Journal of Digital Curation*, 3(2), 2008.
- [115] Kam Woods and Geoffrey Brown. Creating Virtual CD-ROM Archives. *Proceedings of the 5th Annual Conference on Preservation of Digital Objects*, 5, 2008.
- [116] C. P. Write and E. Zadok. Unionfs: Bringing file systems together. *Linux Journal*, (128):24–29, December 2004.
- [117] The Index Data YAZ Toolkit. <http://www.indexdata.com/yaz>. Accessed August, 2010.

Curriculum Vitae

Kam A. Woods

kamwoods@cs.indiana.edu
<http://www.cs.indiana.edu/~kamwoods>
<http://www.linkedin.com/pub/kam-woods/4/92b/115>

668 E. Hillside Drive
Bloomington, IN 47401
(812) 606-1755

- Education**
- Indiana University, Bloomington, IN.
Ph.D. Computer Science (Ph.D. Minor: Computational Linguistics), Defended: August 2010.
 - Baylor University, Waco, TX.
M.S. Computer Science, 2004. Thesis: Automatic Discovery and Classification of Discourse Contingency Relations
 - Swarthmore College, Swarthmore, PA.
B.A., Special Major in Computer Science, May 2002.

- Peer Reviewed Publications**
- K. Woods and G. Brown. 2009. Assisted Emulation for Legacy Executables. To Appear in *Proceedings of the 5th International Digital Curation Conference*.
- G. Brown and K. Woods. 2009. Born Broken - Fonts and Information Loss in Legacy Digital Documents. *Proceedings of the 6th International Conference on Preservation of Digital Objects*.
- K. Woods and G. Brown. 2009. From Imaging to Access - Effective Preservation of Legacy Removable Media. *Proceedings of Archiving 2009*.
- K. Woods and G. Brown. 2008. Virtualization for Preservation of Executable Art. *Proceedings of the 2008 DOCAM Summit and Symposium*.
- K. Woods and G. Brown. 2008. Creating Virtual CD-ROM Collections. *Proceedings of the 5th International Conference on Preservation of Digital Objects*. Also in *International Journal of Digital Curation* Vol. 4, No. 2 2009.
- K. Woods and G. Brown. 2007. Migration Performance for Legacy Data Access. *3rd International Digital Curation Conference*. Also in *International Journal of Digital Curation* Vol. 3, No 2. 2008.
- R. Girju and K. Woods. 2005. Exploring Contingency Discourse Relations. *Proceedings of the 6th International Workshop on Computational Semantics (IWCS-6)*, Harry Bunt (ed.), The Netherlands.

- Conference Presentations**
- Assisted Emulation for Legacy Executables. 5th International Digital Curation Conference.
- From Imaging to Access - Effective Preservation of Legacy Removable Media. Archiving 2009.

Kam A. Woods

Virtualization for Preservation of Executable Art. 2008 DOCAM Summit and Symposium.

Creating Virtual CD-ROM Collections. 5th International Conference on Preservation of Digital Objects.

Migration Performance for Legacy Data Access. 3rd International Digital Curation Conference.

Research interests

Development and evaluation of tools and techniques for long-term digital preservation. Development of systems for high-performance automated format migration, low-risk file format identification, and handling of legacy removable media. Development of risk management tools for access, migration, and accurate rendition of outdated binary formats. Automation of emulation platforms to support continued access to legacy executables. Information retrieval, information extraction.

Academic research projects

- **FDLP CD-ROM virtualization project.** Working with Dr. Geoffrey Brown at Indiana University, I performed primary technical, programming, and analysis tasks on this project, including the development of novel methods for low-risk transfer of legacy materials to archival distributed storage, file distribution and format analysis. Publications based on this work have been accepted to several major international conferences. I assisted in coordination with Indiana University Libraries staff during this project, and supervised multiple undergraduate student programmers.
- **Legacy document font analysis.** With Dr. Geoffrey Brown, continued existing research into the identification, extraction, and use for quality assurance of font data from legacy binary document formats such as Microsoft Word. As part of this project I constructed a novel terabyte-scale dataset subsequently analyzed using existing software libraries and customized code written on-site. (Spring 2009-Present)
- **Format migration and risk analysis.** Continued development of high-throughput, low-risk server-side migration procedures using off-the-shelf open source packages such as OpenOffice and associated APIs. Exploration of automated techniques to identify high-risk components of migrated documents. (Summer 2007-Present, Ongoing)
- **Independent research (Indiana University).** Interlingua research performed with data from the Preliminary Mayan Etymological Dictionary. Automated approaches for ontology construction and mining of semantic information from sparse data. Software prototype and dataset constructed under supervision of Dr. Michael Gasser. (Spring 2007-Fall 2008)

Research and Teaching

- Research Assistant, Indiana University Computer Science Department
May 2007 – present
Long-term digital preservation research with Dr. Geoffrey Brown. In this role I also acted as mentor to undergraduate programmers and data acquisition hires over the course of three years, coordinating development of prototype software systems for data preservation and access.

Kam A. Woods

- Associate Instructor, Indiana University, Computer Science Department
August 2005 – May 2007

Teaching assistant:

- A110: Introduction to Computers and Computing
Taught laboratory sessions emphasizing digital literacy skills, including training students from diverse backgrounds in advanced document creation and management and in basic web development.
- A201: Introduction to Programming
Introduction to programming in Python. Designed laboratory activities, assignments, and tests in course designed primarily for non-Computer Science majors. Across three semesters assisted in improving course design, introducing students to programming techniques using virtual worlds, basic algorithm implementation and analysis, and group programming techniques.

Course instructor:

- A202: Introduction to Programming II
Introductory object-oriented programming in Java. Building on an existing course design, taught an accelerated summer course introducing students to user-defined functions and types, recursion and iteration, parameter-passing mechanisms, and interface design.

*Professional
Service and
Activities*

Member, ACM (2001-present)
Officer, Swarthmore College Computer Society (2000-2002)
Officer, Indiana University Computer Science Graduate Student Association (2005 – 2007)

*Technical
Skills*

- C/C++, Java, .NET, Python, Perl, Scheme, PHP, Shell Scripting
- Long-term digital preservation, Machine Learning, Information Retrieval, and Natural Language Processing