From Imaging to Access - Effective Preservation of Legacy Removable Media

Kam Woods and Geoffrey Brown; Indiana University Department of Computer Science; Bloomington, Indiana

Abstract

We describe how existing media collections can be virtualized through digital copies that are accessible via ordinary workstations. In the model presented, bit-identical images of the original media are served from a distributed file system to provide convenient access on local or remote workstations. Our approach incorporates mature open source libraries to provide high-quality, low-cost image extraction, file format identification, metadata creation, and web-driven access; we include specific examples of custom scripting designed to enhance usability.

Introduction

Over the past 25 years, many research materials have been published on removable digital media such as floppy disks and CD-ROMs. These include approximately 5000 items published by the Government Printing Office (GPO). 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.

The techniques we present 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 [7], 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.

While the principles are simple – create bit-identical copies of original media and export these through a file server – there are significant technical pitfalls that can readily be avoided. We 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.

In previous work we noted several non-catastrophic error conditions that can occur during this process [10]. Here we present 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. We 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.

We provide technical documentation on the automation of other tasks required to facilitate future access. Our methodology for generating high-quality archival metadata is described in detail. We demonstrate the incorporation of existing MARC metadata into METS records, additionally describing related administrative and technical activities using scripted automation and XSLT transforms. We discuss strategies to avoiding common incompatibilities in handling discs mastered for legacy platforms, high-accuracy identification of file formats, and providing web access as a view into a "virtual file system".

In this synthetic hierarchy, a large body of heterogeneous document types such as those present in CD-ROM collections can be browsed quickly and transparently. Our work uses the OpenAFS implementation of the Andrew File System and a directory structure designed to scale appropriately as an archive expands [9]. We provide specific information on setup, administrative and access privileges, and mapping from ISO images stored on disk to a browsing and access view in a web browser.

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. In the remainder of the paper we discuss techniques for imaging legacy media, our experience in building and testing a virtual collection, and the construction of a web-access system with standards-compliant metadata.

Imaging Removable Media

A key premise of our work with the GPO 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.

In this section, we present an approach to extracting bit-faithful images from optical and magnetic media. We 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. Our approach is supported by experiments performed on over 4000 removable media items in the IU Libraries collection. We focus 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.

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 [4]. '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. As we shall discuss, CD-ROMs 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 [5]. Due to a variety of factors in the premastering 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. As we discuss, 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; for this study, we 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 creation.

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 we 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 and length of each file individually. By performing this type of filesystem walk with a simple C program using the `libiso9660` library, we discovered approximately 8% of the images we 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 an experiment with 86 CD-ROMs whose images were truncated by Windows, we were able to extract a complete ISO image for 81 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. We expect the rate of failure to read all the bits of CD-ROMs to be under 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.

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. We can therefore build a complete case 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` package[1] and standard GNU `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 1. *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.[2] 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.

Building the Collection

Over the past 3 years, a single part-time work-study 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. In this section we describe our process as well as some lessons learned.

The key interface between the research team and AF consisted of a networked file system, where she deposited images that she created, 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 [1]. 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. Subsequently, we used the key to access the MARC record of each item through the Indiana University library z39.50 server.

[1] http://www.chrysocome.net/dd

[2] http://support.microsoft.com/kb/138434

Initially, we populated the database 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 as "outsiders" we have no 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 our 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 [9]. 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.

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.

The most important lessons we learned were the issues associated with creating bit faithful images, the need to regularly check created images for quality, and the need for automated reporting structures. We had a large number of bad images – 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.

Automated reporting scripts were used to track progress while constructing the collection. Initially, we used the ISO verification tool `isovfy` 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.

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.[3] 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 in Figure 2.

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
Total items tested, each category:	1382

Figure 2. *Testing access. Note numeric overlap in the application tests; for example, access to the information on a given ISO may require both a browser and additional binaries.*

KM has 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.

Accessing the Collection

In this section we describe an approach to building a web site to enable searching and browsing a collection of media images. 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 3. All collection specific information in our system 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.

Our 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.

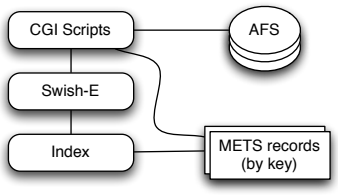

Figure 4. *Metadata handling*

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 5 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.[4]

The web service processes METS records. We use standard indexing tools to process these METS records and all displayed bibliographic information is dynamically generated from the METS records. We update the collection 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

[3] http://www.fdlp.gov/administration/computers/244-mtr

[4] The 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`.

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

Figure 3. *Namespace*

```
   -- (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 5. *Example METS Record*

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 [6]. Each entry in our database included an internal IU catalog key used to generate a z39.50 query. The retrieved MARCXML records were converted to MODS with a lightly modified version of the Library of Congress MARCXML to MODS stylesheet.[5] These records were then processed using Perl scripts to build METS documents containing links to the media images and to build a searchable index.

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.

[5] http://www.loc.gov/standards/mods/v3/MARC21slim2MODS3-3.xsl

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, we had 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 3 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 [11]. As noted previously, we use the `libsharedmime` library 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.[6] These implementations are fast, platform-agnostic, and provide the breadth of coverage required to deal with large, heterogeneous collections of files.

Discussion

This work presents simple, low-cost techniques intended to assist archival institutions with legacy media holdings in the low-risk creation of virtual collections. We discuss CD-ROM and floppy disk verification via walking the filesystem directly, and show that many commonly used commercial tools fail to obtain a correct sector count for incorrectly premastered images, or introduce additional hidden error.

We demonstrate the numerous advantages of virtual collections, including distributed storage and the facilitation of network access. For the majority of images, a filesystem mount is not required; the contents can be accessed directly using fast, low-overhead open source libraries. For legacy images requiring an installation procedure, use of a virtual machine avoids contamination of the local environment.

We use the interoperable METS metadata standard to reformulate existing bibliographic MARC records and construct a searchable index for web-based access, and discuss the use of a mature open source file format identification procedure to identify individual items within extracted images.

[6] http://www.memecode.com/libsharedmime.php

Our approach is a blueprint for creating high-quality, low-risk archives of CD-ROM and floppy disk images with an emphasis on access. We show that automated or scripted error detection can reduce risk in transfer of archival data from legacy media to low-cost, high-reliability redundant disk arrays. We provide a method for processing a collection of legacy media and associated metadata and creating a complete web-access solution using freely available open source projects and less than 3000 lines of custom scripting.

References

[1] Virginia Allen. Automated library inventory using bar codes. In *Proceedings of the 23rd IAMSLIC Conference*, page 5, Charleston, SC, USA, 1987. IAMSLIC.

[2] Doe & Moffitt Libraries CD Archive. `http://sunsite.berkeley.edu/wikis/datalab/Data/CdArchive`. Accessed December 2008.

[3] CD-ROM Redigitization Project Demonstration. `http://www.archive.org/details/code4lib.conf.2008.lightningtalk.CDRIP`. Accessed December 2008.

[4] Care and Handling of CDs and DVDs: A Guide for Librarians and Archivists. `http://www.clir.org/pubs/reports/pub121/contents.html`. Accessed December 2008.

[5] EMCA. Standard EMCA-119 Volume and File Structure of CDROM for Information Exchange, 1987.

[6] Index Data. The YAZ Programmers' Toolkit, 2008.

[7] Sudoc Virtualization Project at Indiana University. `http://www.cs.indiana.edu/svp/`. Accessed December 2008.

[8] Raymond A. Lorie. A methodology and system for preserving digital data. In *Proceedings of the second ACM/IEEE-CS joint conference on Digital libraries*, pages 312–319. ACM Press, 2002.

[9] OpenAFS Distributed File System. `http://www.openafs.org/`. Accessed December 2008.

[10] K. Woods and G. Brown. Creating Virtual CD-ROM Archives. *Proceedings of the 5th Annual Conference on Preservation of Digital Objects*, 5, 2008.

[11] K. Woods and G. Brown. Migration Performance for Legacy Data Access. *International Journal of Digital Curation*, 3(2), 2008.

Author Biographies

Kam Woods is a Research Assistant and Ph.D candidate in Computer Science at Indiana University Bloomington. He holds a BA with a major in Computer Science from Swarthmore College, and an MS degree in Computer Science from Indiana University. His research focuses on long-term digital preservation software development.

Geoffrey Brown is a Professor of Computer Science at Indiana University, Bloomington. He received the BS in engineering from Swarthmore College, MSEE from Stanford University, and PhD from the University of Texas at Austin in 1987. He taught at Cornell from 1987-1997 and worked as a research scientist at Hewlett Packard Laboratories, and architect at several networking startups. His research interests include verification and design of digital systems, and the preservation of digital documents.